# Computational Analysis of Gene Regulatory Networks

D I S S E R T A T I O N

zur Erlangung des akademischen Grades

d o c t o r  r e r u m  n a t u r a l i u m
(Dr. rer. nat.)

im Fach Biophysik

eingereicht an der
Mathematisch-Naturwissenschaftlichen Fakultät I
Humboldt-Universität zu Berlin

von

HERRN DIPL.-PHYS. HENDRIK HACHE
geboren am 21. April 1977 in Berlin

Präsident der Humboldt-Universität zu Berlin
Prof. Dr. Dr. h.c. Christoph Markschies

Dekan der Mathematisch-Naturwissenschaftlichen Fakultät I
Prof. Dr. Lutz-Helmut Schön

Gutachter:

1. Prof. Dr. Edda Klipp, Humboldt-Universität zu Berlin
2. Prof. Dr. Hans Lehrach, Max-Planck-Institut für Molekulare Genetik
3. Prof. Dr. Joachim Selbig, Universität Potsdam

Eingereicht am: 22. Juni 2009
Tag der mündlichen Prüfung: 03. November 2009

# Abstract

Gene regulation, the coordinated control of gene expression, is accomplished mainly by the interplay of multiple transcription factors. This gives rise to highly complex and cell-type specific, interwoven structures of regulatory interactions summarized in gene regulatory networks. Computational analysis of gene regulatory networks can be classified in forward modeling approaches and reverse engineering methods. The goal of forward modeling is the prediction of gene expression dynamics based on gene regulatory networks and perturbations thereof. Such forward modeling approaches need comprehensive information on network structure, kinetic laws, and kinetic parameters. The goal of reverse engineering methods is the reconstruction of the underlying network topology from experimental data.

In this thesis, I address both approaches of computational analysis of gene regulatory networks. The first part of this thesis is about the Web application GEne Network GEnerator (GeNGe, `http://genge.molgen.mpg.de`) which I have developed as a framework for automatic generation of gene regulatory network models. GeNGe provides a user-friendly interface with functionalities for forward modeling. I have developed a novel algorithm for the generation of network structures featuring important biological properties. In order to model the transcriptional kinetics, I have modified an existing nonlinear kinetic. This new kinetic is particularly useful for the computational setup of complex gene regulatory models. GeNGe supports also the generation of various *in silico* experiments for predicting effects of perturbations as theoretical counterparts of biological experiments. Moreover, GeNGe facilitates especially the collection of benchmark data for evaluating reverse engineering methods. The validation with such artificial data supports the development and testing of reverse engineering algorithms.

The second part of my thesis is about the development of GNRevealer, a method for reverse engineering of gene regulatory networks from temporal

data. This computational approach uses a neural network together with a sophisticated learning algorithm (backpropagation through time). Specialized features developed in the course of my thesis include essential steps in reverse engineering processes such as the establishment of a learning workflow, discretization, and subsequent validation. Additionally, I have conducted a large comparative study using six different reverse engineering applications based on different mathematical backgrounds. The results of the comparative study highlight GNRevealer as best performing method among those under study.

# Zusammenfassung

Genregulation bezeichnet die geregelte Steuerung der Expression von Genen durch das Zusammenspiel einer Vielzahl von Transkriptionsfaktoren. In ihrer Gesamtheit basiert die Regulation von Genen auf hoch komplexe und zellspezifische genregulatorische Netzwerke. Computergestützte Analysemethoden solcher Netzwerke können in Vorwärtsmodellierung und *Reverse Engineering* Ansätze unterteilt werden. Das Ziel der Vorwärtsmodellierung ist die Vorhersage von Genexpressionsprofilen auf der Basis von genregulatorischen Netzwerken unter Berücksichtigung von spezifischen und unspezifischen Störungen dieses Systems. Dieser Ansatz der Vorwärtsmodellierung benötigt umfangreiche Informationen über die Struktur des Netzwerks, der kinetischen Gesetzmäßigkeiten und der zugehörigen kinetischen Parameter. Das Ziel von *Reverse Engineering* Methoden ist hingegen die Rekonstruktion von genregulatorischen Netzwerktopologien auf Basis experimenteller Daten.

Im Rahmen meiner Doktorarbeit beschäftigte ich mich mit diesen beiden Ansätzen der computergestützten Analyse von genregulatorischen Netzwerken. Der erste Teil dieser Arbeit beschreibt die Entwicklung der Web-Anwendung GEne Network GEnerator (GeNGe, `http://genge.molgen.mpg.de`). Hierbei handelt es sich um ein System für die automatische Erzeugung von genregulatorischen Netzwerken. GeNGe ist über eine frei zugängliche und benutzerfreundliche Oberfläche zu bedienen und stellt viele Funktionen für die Vorwärtsmodellierung zur Verfügung. Hierfür entwickelte und implementierte ich einen neuartigen Algorithmus für die Generierung von Netzwerkstrukturen die wichtige Eigenschaften biologischer Netzwerke zeigen. Für die dynamische Beschreibung der Transkription modifizierte ich eine nicht-lineare Kinetik. Diese neue Formulierung der Kinetik eignet sich besonders für die Erstellung von komplexen genregulatorischen Modellen am Computer. Desweiteren unterstützt GeNGe die Durchführung verschiedener *in silico* Experimente, um theoretische Aussagen über den Einfluss von Störungen des Systems treffen zu können. Darüber hinaus erleichtert GeNGe die Erzeugung solcher

Daten, die für die Evaluierung und Validierung von *Reverse Engineering* Methoden herangezogen werden können.

Der zweite Teil meiner Doktorarbeit beschreibt die Entwicklung der Anwendung GNRevealer. Hierbei handelt es sich um eine Methode zur Rekonstruktion von genregulatorischen Netzwerken auf Basis von Messungen der Genexpression zu verschiedenen Zeitpunkten. Diese Methode verwendet ein neuronales Netz zusammen mit einem passenden Lernalgorithmus (*backpropagation through time*). Modifizierungen, welche notwendig für die Anwendung im *Reverse Engineering* Bereich sind, wurden von mir entwickelt, wie z.B. die Etablierung verschiedener Schritte eines vollständigen Lernprozesses, die Diskretisierung der Ergebnisse und anschließende Validierungen. Im letzten Teil dieser Arbeit beschreibe ich eine große Vergleichsstudie, in der sechs verschiedene *Reverse Engineering* Anwendungen von mir miteinander verglichen wurden. Diese Untersuchung hebt GNRevealer als geeignetste Anwendung aller getesteten Methoden hervor.

# Acknowledgement

I am very grateful to Prof. Dr. Edda Klipp for being my adviser and for reviewing my thesis.

I would like to thank Prof. Dr. Hans Lehrach for giving me the opportunity to dive into the exciting field of reverse engineering. It has been a pleasure to participate in the research of his department at the Max Planck Institute for Molecular Genetics and its interdisciplinary environment.

Moreover, I wish to express my sincere appreciation to Prof. Dr. Joachim Selbig for reviewing this thesis.

I give my sincere gratitude to Dr. Ralf Herwig for his guidance, criticism, patience, encouragements, and his support during developing and completing this work. He provided me valuable perspectives and his keen sense of the key aspects was highly beneficial.

I would like to sincerely acknowledge Dr. Mathias Steinfath and Dr. Dirk Repsilber for inviting me to their exciting workshop about integrative network analysis and encouraging me to contribute to its special issue. I also want to thank Dr. Daskalaki for inviting me to submit a chapter for her wonderful book.

I am indebted to Dr. Christoph Wierling who has provided particular assistance in numerous ways. He is a congenial colleague and has contributed towards my understanding and thoughts in the research and beyond.

Furthermore, I want to express my gratitude to my colleagues from the Bioinformatics group and the newly established Systems Biology group of the department of Prof. Lehrach for providing an enjoyable team. I also want to highlight the wonderful atmosphere with all the fellow students at the institute. It has been a pleasure working and enjoying the time with them.

I particularly extend my thanks to Marcus Albrecht, Rebecca Crane, Felix Dreher, Ralf Herwig, Christoph Wierling, Wasco Wruck, Reha Yildirim for proof-reading and commenting on the manuscript of my thesis. Moreover, I thank Konstantin Pentchev for providing me nice looking network graphs.

My special gratitude goes to my dear family whose love, encouragement, and support mean so much to me.

Lastly, I offer my regards to all of those who supported me in any respect during the completion of my thesis.

This work was funded by the Max Planck Society.

# Contents

**Abstract** iii

**Zusammenfassung** v

**Acknowledgement** vii

**Contents** ix

**List of Figures** xiii

**List of Tables** xv

**1 Introduction** 1

1.1 Biological Background . . . . . 2

1.1.1 Gene Expression . . . . . 2

1.1.2 Transcriptional Regulation . . . . . 4

1.1.3 Experimental Methods for Studying Gene Regulatory Networks . . . . . 6

1.2 Databases and Resources . . . . . 7

1.3 Mathematical Background . . . . . 10

1.3.1 Gene Regulatory Networks . . . . . 10

1.3.2 Topological Characterization . . . . . 12

1.4 Reverse Engineering Methods . . . . . 17
1.4.1 Neural Networks . . . . . 19
1.5 Objectives and Outline . . . . . 21

**2 Forward Modeling: Generation of Artificial Benchmark Data with GeNGe 23**

2.1 Modeling of Transcriptional Regulation . . . . . 24
2.1.1 Simplified Gene Regulation Model . . . . . 24
2.1.2 State-Of-The-Art in Forward Modeling Gene Regulatory Networks . . . . . 31
2.2 GEne Network GEnerator . . . . . 36
2.2.1 Network Level . . . . . 37
2.2.2 Kinetic Level . . . . . 43
2.2.3 Simulation Level . . . . . 46
2.2.4 Simulation Results . . . . . 47
2.2.5 Forward Modeling examples . . . . . 48
2.3 Network Properties . . . . . 54

**3 GNRevealer – a Neural Network Approach for Reverse Engineering 57**

3.1 Model . . . . . 57
3.2 Learning Algorithm . . . . . 60
3.2.1 Discretization . . . . . 66
3.3 Performance . . . . . 69
3.4 Comparison to Linear Approach . . . . . 72
3.5 Application to Network Motifs . . . . . 76
3.6 Extensions . . . . . 77

**4 Comparison of Reverse Engineering Methods 81**

4.1 Computational Methods . . . . . 81
4.1.1 Relevance Networks . . . . . 82

4.1.2 Bayesian Networks . . . . . 84
4.1.3 Graphical Gaussian Models . . . . . 87
4.1.4 State Space Models . . . . . 87
4.2 Validation Measures . . . . . 88
4.3 Simulation Data . . . . . 90
4.4 Performance Results . . . . . 91
4.5 Discussion of Performance Results . . . . . 96

**5 Discussion** **99**
5.1 Forward Modeling . . . . . 100
5.2 Reverse Engineering . . . . . 104
5.3 Comparative studies . . . . . 108

**A Network Motifs** **111**
A.1 Auto-Regulation . . . . . 112
A.2 Single-Input Motif . . . . . 112
A.3 Feed-Forward Loop . . . . . 113
A.4 Multi-Input Motif . . . . . 113
A.5 Multi-Component Loop . . . . . 114
A.6 Regular Chain . . . . . 114

**B Transcription Function** **115**

**C Technical Aspects of GeNGe** **121**
C.1 Behind the Web Interface . . . . . 121
C.2 Scaling Behavior . . . . . 122

**D BPTT Algorithm** **125**
D.1 Additive Model . . . . . 125
D.2 Multiplicative Model . . . . . 129

**E Classification Matrix** **131**

**F Publications** **133**

**Abbreviations** **135**

**Bibliography** **139**

**Selbstständigkeitserklärung** **157**

# List of Figures


1.1 Illustration of gene expression in eukaryotic cells . . . . . 3
1.2 Illustration of transcriptional regulation . . . . . 5
1.3 Representations of a gene regulatory network . . . . . 11
1.4 Different perceptron models . . . . . 20
1.5 Framework for modeling gene regulation . . . . . 21

2.1 Simplified model of gene regulation . . . . . 26
2.2 Illustration of joint interaction sets . . . . . 28
2.3 Response characteristic of various transcription models . . . . . 32
2.4 Response characteristic of various transcription models (cont.) . 33
2.5 Flowchart of the simulation process in GeNGe . . . . . 37
2.6 Network level of GeNGe . . . . . 38
2.7 Motif network generation . . . . . 39
2.8 Partial graph of developmental gene regulatory network . . . . 42
2.9 Kinetic level of GeNGe . . . . . 44
2.10 Simulation level of GeNGe . . . . . 47
2.11 Simulation results in GeNGe . . . . . 48
2.12 Toggle switch model . . . . . 49
2.13 Oscillatory model . . . . . 51
2.14 All single knock-outs in developmental network . . . . . 53

2.15 Generated networks . . . . . 56

3.1 Neural network representation . . . . . 59
3.2 GNRevealer workflow . . . . . 61
3.3 Stop criterion . . . . . 65
3.4 Determination of an optimal threshold for discretization . . . . 67
3.5 Classification matrix . . . . . 68
3.6 Developmental networks . . . . . 71
3.7 Reconstruction on data generated with neural network . . . . . 72
3.8 Fluctuations in results . . . . . 74
3.9 Scaling behavior of GNRevealer . . . . . 75
3.10 Comparison GNRevealer and linear approach . . . . . 75
3.11 Motif reconstruction properties . . . . . 77
3.12 Modified errors . . . . . 79
3.13 Reverse Engineering with modified error function . . . . . 80

4.1 Performances of applications . . . . . 94
4.2 Performances of applications (cont.) . . . . . 95

5.1 Proposed parameter prior . . . . . 107

A.1 Motif auto-regulation . . . . . 112
A.2 Motif single-input motif . . . . . 112
A.3 Motif feed-forward loop . . . . . 113
A.4 Motif multi-input motif . . . . . 113
A.5 Motif multi-component loop . . . . . 114
A.6 Motif regular chain . . . . . 114

C.1 Connection to PyBioS . . . . . 122
C.2 Scaling behavior of GeNGe: network generation . . . . . 123
C.3 Scaling behavior GeNGe: Data Generation . . . . . 124

E.1 Classification matrix and definitions . . . . . 132

# List of Tables

1.1 Several Gene Regulation Databases . . . . . 8

2.1 Response Characteristics . . . . . 30
2.2 Units in GeNGe . . . . . 45
2.3 Topological measures of generated networks . . . . . 55

3.1 Comparison of different threshold optimizations . . . . . 69
3.2 Reconstructed regulations in developmental network. . . . . 73

4.1 Reverse engineering applications in the comparative study . . . 82
4.2 Discretization thresholds for different types of measures. . . . . 91
4.3 Performances of applications . . . . . 93

# CHAPTER 1

# Introduction

The coordinated control of gene expression is accomplished by the interplay of multiple transcription factors and is one of the fundamental levels in the dynamics of cellular processes. Basic concepts of the regulation of transcription processes through DNA-binding factors have been first developed over forty years ago by Monod and Jacob (1961). Recently, it has been found that 5-10% of the total coding capacity of metazoans is dedicated to proteins that regulate transcription (Levine and Tjian, 2003). However, it is not the isolated and individual properties of each constituent, rather the entirety and interactions between the components which determine the behavior of the regulating system (Regenmortel, 2004, Mazzocchi, 2008). This concerted regulation of gene expression gives rise to a structure of regulatory interactions called gene regulatory network. The temporal and spatial transcriptional regulation is responsible for the complexity and diversity in the course of biological evolution and adaptation (Nguyen and D'haeseleer, 2006). Therefore, knowledge of the underlying gene regulatory networks is of central importance in molecular biology.

Multiple data mining techniques have been applied to gather information about the dynamic processes of transcriptional regulation. Data-driven approaches that deduce network structure from experimental (temporal) observations are classified as reverse engineering methods. There are many analogous terms for reverse engineering in the context of gene regulation, such as recovering, inferring, deciphering, elucidating, identifying, revealing, reconstructing, uncovering, deducing, or unraveling of gene regulatory networks.

All of which refer more or less to the same topic. They aim to reverse engineer networks of interactions between cellular entities from high throughput biological data all at once rather than one interaction at time. To tackle this problem, a battery of reverse engineering methods has been developed based on different mathematical approaches. These methods are adapted to different problem domains, such as perturbation and time series analyses, and have to cope with several difficulties, such as small, noisy, and incomplete datasets.

Besides the algorithmic developments, the actual assessment of methods performance remains a challenge, primarily due to the lack of experimental benchmark data. The performance of individual methods is poorly understood and validation of algorithmic performances is still missing to a large extent. However, such systematic validation is crucial since it shows strengths and weaknesses of the individual methods and their suitability for the specific problem domain. Availability of experimental data, with a few exceptions like the network described by Davidson et al. (2002), is still the major bottleneck for gene regulatory network reconstruction. Thus, generating simulated data derived from theoretical considerations is still the method of choice for constructing benchmark data sets.

## 1.1 Biological Background

### 1.1.1 Gene Expression

In almost all biochemical processes in every cell of a living organism proteins or protein complexes are involved. For instance, proteins are essential for enzymatic catalyzation of biochemical reactions, transmission of signals via the cell communication systems, induction of conformational changes of other molecules, identifying and neutralizing foreign objects in cells as an immune response, maintaining cell shape in the cytoskeleton, regulating transcriptional processes of genes, and many other functions. The instruction for building proteins is stored in certain regions of DNA that define genes. This sequence information is transcribed and translated into polypeptides which in turn are transformed into functional proteins. The process of transcription and translation in eukaryotic cells involves several steps that are illustrated in Figure 1.1. Besides protein-coding genes, the DNA encodes also information of functional RNAs, such as ribosomal RNA (rRNA), transfer RNA (tRNA), and micro RNA (miRNA), which are not translated into proteins but are involved in several cell functions. The whole process of transcription and translation including all intermediate steps is termed gene expression.

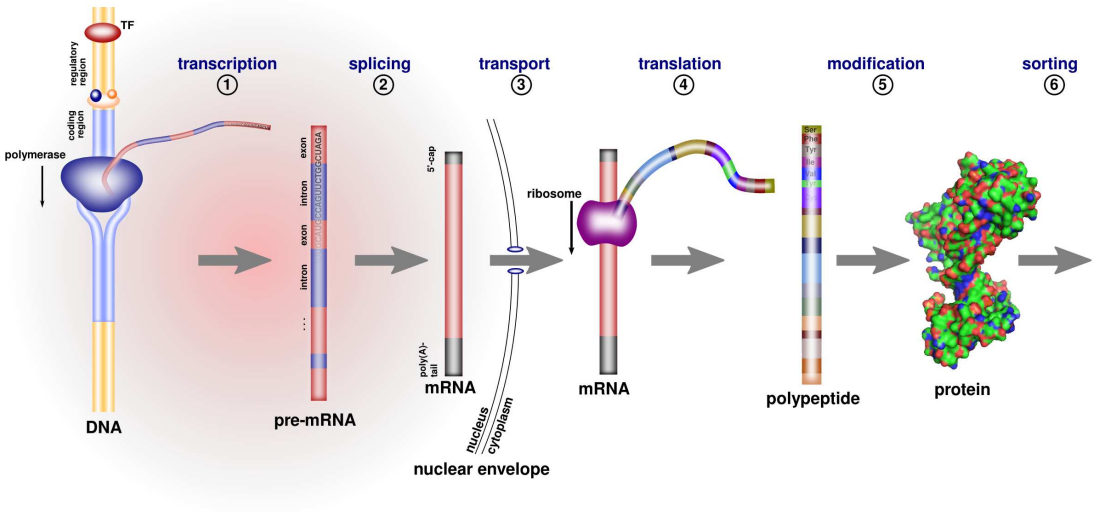



Figure 1.1 | **Illustration of gene expression in eukaryotic cells.** Several steps are involved in the process of gene expression starting with transcribing genes and leading, ultimately, to a protein.

Gene expression starts with transcription (① Figure 1.1), i.e., the synthesis of a messenger RNA (mRNA) from a DNA template. The most important parts of DNA are the coding and regulatory regions. DNA is composed of a sequence of the four nucleotides adenine, cytosine, guanine, and thymine. Coding sequences of DNA which serve as templates are enzymatically copied into a complementary mRNA by the RNA polymerase. A collection of proteins called transcription factors favor or prevent the binding of the enzyme RNA polymerase and the initiation of transcription by binding to the corresponding regulatory region of the gene. Several other proteins prepare DNA for a successful binding of RNA polymerase near the transcription initiation site upstream of the gene. Once the transcription is initiated, an mRNA copy can be synthesized. As transcription proceeds, RNA polymerase unwinds the DNA duplex strand and traverses along one of the DNA strands. It uses base pairing complementarity with the DNA template to create an mRNA copy with the exception that instead of the nucleotide thymine uracil is used in RNAs. The transcription is terminated, if the polymerase reaches the terminator sequence of the gene.

The primary mRNA is called precursor mRNA (pre-mRNA) since it contains coding sequences flanked by non-coding sequences. Non-coding sequences are removed (introns) in the next step of gene expression and coding sequences are joined (exons) sequentially (② in Figure 1.1). This splicing process is catalyzed by the spliceosome. Several distinctive mature RNAs can be processed from the same pre-mRNA by joining different sets of exons. This is called alternative splicing and leads finally to different proteins having possibly dif-

ferent functions. Additionally, pre-mRNA is processed by adding non-coding sequences, a 5′ cap to the flanking 5′ region and a series of adenine nucleotides (poly(A)-tail) at the 3′ end to stabilize the mRNA.

After accomplished splicing, the mature mRNA is transported through pores in the nuclear envelope to the cytoplasm (③ in Figure 1.1), where it is translated into the final protein during the next steps. Translation (④ in Figure 1.1) of the nucleotide sequences into an amino acid sequence is performed by ribosomes in the cytosol. Triplets of nucleotides are read sequentially from the mRNA and are translated into amino acids. A triplet, also called codon, encodes one of the twenty amino acids[1]. Appending amino acids to the nascent protein is repeated until a stop codon is reached. The translation is terminated and the ribosome releases the mRNA and the polypeptide. Several mechanisms exist in the cell to fold the polypeptide into a three-dimensional protein structure (⑤ in Figure 1.1). Finally, proteins are directed to their final destination in the cell, where they accomplish their destined function (⑥ in Figure 1.1).

### 1.1.2 Transcriptional Regulation

Several genes are expressed continuously since they produce proteins involved in basic cell functions. However, the majority of genes are expressed only at a certain time and in a particular cell type in a coordinated fashion. The amount of proteins present in a cell is decisively regulated by the transcriptional control machinery in each single cell and even possibly coordinated with neighbor cells via signals. This machinery drives the processes of cellular differentiation, development, and morphogenesis, leading to different cell types with different gene expression profiles.

Gene transcription is a remarkably elaborate biochemical process that can be regulated at any level during gene expression. For instance, transcriptional regulation takes place by chemical and structural modifications of DNA, manipulation of the stability of the gene products, modulation of the binding affinity and elongation rate of the RNA polymerase, and post-transcriptional and post-translational modifications. However, the regulation of the transcription rate by physical interactions of transcription factors with the DNA is the predominant form and consequently of crucial importance and interest. Transcription factors bind to regulatory regions of genes and activate, enhance, or inhibit the initiation of the transcription. Once the correct transcription factors are bound, the RNA polymerase binds to the promoter (Figure 1.2) and initiates the transcription. A promoter is a region of DNA located upstream

[1] Each codon leads to a certain amino acids, but multiple codons may encode the same amino acid.

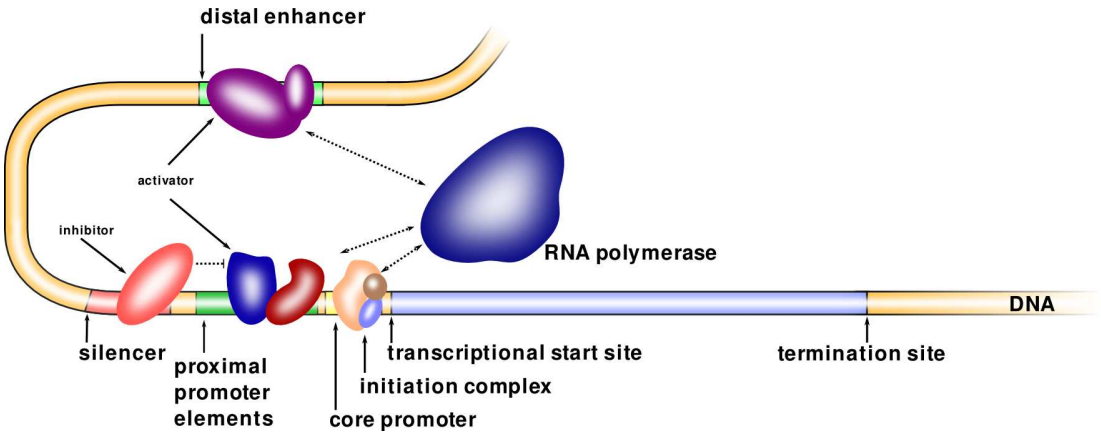



Figure 1.2 | **Illustration of transcriptional regulation.** Several transcription factors bind to sequence-specific regions on the DNA to inhibit or enhance the binding of the RNA polymerase at a promoter where other, general transcription factors are required to recruit a pre-initiation complex with the RNA polymerase. Dotted lines indicate physical interactions.

near the transcription start site of a gene and provides a binding site for RNA polymerase. The next polymerase cannot bind to the promoter and initiates a further transcription before the currently transcribing polymerase has cleared a certain amount of DNA (de Leon and Davidson, 2009). Hence, the transcription rate shows a saturation to an upper limit determined by the translocation rate of the RNA polymerase.

There are general but diverse transcription factors required for the formation of a pre-initiation complex for promoter recognition of the RNA polymerase. Furthermore, other transcription factors bind to short and highly specific DNA sequences and mediate gene-selective transcriptional activation or repression. DNA regions where transcription factors can be bound in order to enhance transcription are called enhancers, while silencers are DNA regions bound by transcription factors in order to prevent RNA polymerase binding to its promoter, an effect that in turn inhibits expression. Enhancers are composed of multiple binding sites for different regulatory factors and are located upstream or downstream of the coding region, as well as within introns. Furthermore, it is observed that enhancers act over distances of 100 kp or more from the transcription start site resulting in long-range regulations. In contrast, repressors must bind maximal 50-100 bp away of an activator or promoter in order to inhibit expression (Levine and Tjian, 2003).

Despite many investigations of gene regulation, the detailed mechanisms by which individual genes are turned on and off is still inaccurately known (Lemon and Tjian, 2000). The complexity of the general machinery of the eukaryotic transcriptional process surprised the biologists. Especially the numerous and intricate layers of control imposed by the diversification of co-

activators and co-repressors were unexpected.

Genes are regulated by a complex combinatorial interplay of transcription factors. These factors are also gene products and their production is controlled by further transcription factors. These connections between transcription factors and genes establish an elaborate gene regulatory network. The architecture of this network determines the temporal order of specification events in organisms. Identifying regulatory interactions between transcription factors and genes by computational methods especially the identification of directed links between genes, is a focus of this thesis.

### 1.1.3 Experimental Methods for Studying Gene Regulatory Networks

A global picture of transcriptional activities can be obtained by gene expression analysis using DNA array platforms. This is the most prominent experimental technique to measure transcriptional activities for thousands of genes in parallel. Differences in the transcriptome between different tissues or between different disease or treatment states can be identified. Furthermore, changes in the expression levels of genes under study can be monitored over time with DNA arrays. With the routine practical use of such experimental techniques sophisticated computational data analysis strategies, such as reverse engineering, have been developed for studying gene regulatory networks.

The development of DNA array techniques dates back to the late 1980s. Since that time, several array platforms have been developed and have dramatically accelerated many types of investigations in the following decades. The principle design of all array platforms are similar where fragments of DNA serve as probes (Klipp et al., 2005, pp. 289–291). A large number of probes are immobilized or synthesized at predetermined positions on a solid surface. Labeled complementary DNA (cDNA) derived from RNA samples from the tissue of interest is hybridized to specific probes, i.e., they form double stranded DNA hybrids by complementary base pairing. Afterwards, the array is scanned and the amount of bound labeled material is quantified. The signal intensities are transformed into numerical values and are assigned to each probe. These values reflect the abundance of target molecules corresponding to the respective probe and changes in values are interpreted as concentration changes in the biological sample.

Platform technologies differ in the material of the array surface, the type and length of probes, probe density, and labeling procedure of target molecules. Prominent array designs are macroarrays (Poustka et al., 1986, Lehrach et al., 1990, Lennon and Lehrach, 1991) and microarrays (Schena et al., 1995, DeRisi

et al., 1996; 1997). Moreover, several commercial DNA arrays have been established, such as Affymetrix (Lockhart et al., 1996, Wodicka et al., 1997), Agilent (Hughes et al., 2000; 2001), and Illumina (Gunderson et al., 2004, Kuhn et al., 2004). Many comprehensive studies have been conducted using such technologies for genome-wide screenings in parallel quantitative gene expression, for instance, the identification of cell cycle genes in *Saccharomyces cerevisiae* (Spellman et al., 1998, Cho et al., 1998) and in human HeLa cells (Whitfield et al., 2002), temporal expression mapping of central nervous system (Wen et al., 1998), and construction of a compendium of expression profiles corresponding to diverse mutations and treatments (Hughes et al., 2000). However, several comparative studies of different DNA array platforms revealed considerable divergence across different platforms (Tan et al., 2003) and probe-specific factors which influence measurements differently (Kuo et al., 2002).

Spatial and temporal changes in the transcriptome occur naturally or are forced by external stimuli, e.g., by chemical treatments or genetic transformation of cells. A newly developed technique for silencing specific genes uses the phenomenon of RNA interference (RNAi) which is part of a natural defense system of eukaryotes against parasitic genes and the transcriptional control. The effect of sequence-specific down-regulation of certain genes by injection of double-stranded RNA (dsRNA) into the nematode *Caenorhabditis elegans* was discovered by Fire et al. (1998). dsRNA prevents complementary mRNA of being translated into proteins. The RNAi machinery can be used to artificially knock-down the corresponding gene to a certain suppression level, typically between 10% and 70% (Holen et al., 2002) and, hence, perturb the system significantly compared to the untreated system. Furthermore, RNAi knock-down experiments coupled with DNA microarrays allow the detection of transcriptional regulatory influences of silenced transcription factors on their target genes (Berns et al., 2004, Babaie et al., 2007).

## 1.2 Databases and Resources

The collection of biological data in an electronic and computational amenable form is one of the major tasks in current bioinformatics. Using existing knowledge together with new experimental results is fundamental for an improved understanding of biological processes and, hence, it is indispensable to have a fast and easy access to collected, presorted, and filtered data arisen from various biological sources. These sources are for instance small and large scale measurements of gene expression, data of protein and metabolite abundances, sequence information of DNA, RNA, or proteins, identified transcription factor binding sites, revealed transcriptional regulators, protein-protein interactions,

Table 1.1 | **Several gene regulation databases.**

| Database | URL | References |
|---|---|---|
| YEASTRACT | `http://www.yeastract.com` | Monteiro et al. (2008) |
| RegulonDB | `http://regulondb.ccg.unam.mx/` | Gama-Castro et al. (2008) |
| TRED | `http://rulai.cshl.edu/TRED` | Jiang et al. (2007) |
| TRANSFAC | `http://www.gene-regulation.com/` | Matys et al. (2006) |
| JASPAR | `http://jaspar.genereg.net/` | Bryne et al. (2008) |
| PubGene | `http://www.pubgene.org` | Jenssen et al. (2001) |

kinetics of enzymatic activities including parameter values, and many others (Wierling et al., 2007).

Several of the databases listed in the *Nucleic Acids Research* online Molecular Biology Database Collection[2] (Galperin and Cochrane, 2009) are focusing on transcriptional regulation (Table 1.1). Two of them are species specific; Yeast Search for Transcriptional Regulators And Consensus Tracking (YEASTRACT) and RegulonDB are specialized on *Saccharomyces cerevisiae* and *Escherichia coli*, respectively. YEASTRACT (Monteiro et al., 2008) is a curated repository of more than 30 000 regulatory associations between transcription factors and target genes. Additionally, the description of 284 specific DNA binding sites for 108 characterized transcription factors is included. All information is based on more than 1000 bibliographic references from an exhaustive literature analysis. The major part of the described regulatory associations is gathered from publications about ChIP and microarray analyses. An identified association in the database is either of type "documented", i.e., a transcriptional regulatory effect which is experimentally discovered, or of the type "potential", i.e., a supposed regulation based on a match of the transcription factor consensus binding site on a subsequence of the target gene promoter.

The database RegulonDB (Gama-Castro et al., 2008) is focused on the well studied *E. coli* K12 bacterium and is one of the largest database of curated knowledge of transcriptional regulations. It contains comprehensive information of transcriptional regulation on different levels, such as transcription factors, small RNAs (sRNAs), and various sequence and functional information about operons, e.g., promoters, transcription factor binding sites, and terminators. Recently, data of attenuators, riboswitches, and sRNA targets have been included. The user-friendly interface including high quality visualizing tools makes it easy to extract and inspect provided information and identify relationships between different objects. The evidences associated to all objects in the RegulonDB are classified as strong or weak. This is based on the type and

[2]The Molecular Biology Database Collection of currently more than 1100 databases can be found at `http://www.oxfordjournals.org/nar/database/a/`.

confidence level of the corresponding experiment supporting the object and its relationships. The current release 6.3 contains information about 4500 genes, 1700 promoters, 167 transcription factors and more than 2300 regulatory interactions collected from nearly 4000 publications.

A transcriptional regulation database for higher eukaryotes is the Transcriptional Regulatory Element Database (TRED; Jiang et al., 2007). It contains collected information of mammalian *cis*-regulatory elements[3], such as promoters and binding motifs, and *trans*-regulatory elements[4], such as transcription factors. The database provides a relatively complete genome-wide promoter annotation for human, mouse, and rat where a quality level (one of possibly six levels) based on the reliability of the supporting evidences is assigned. Furthermore, most of the transcription factor binding and regulation information is gathered from exhaustive literature curation. Binding motifs are mapped on promoters of the target genes and binding quality levels (one of possibly four levels) are assigned based on definitiveness of the binding evidence. TRED is primarily focused on target genes of cancer-related transcription factors and contains currently around $\sim$ 11 700 target genes ($\sim$ 7500 in human, $\sim$ 2700 in mouse, and $\sim$ 1500 in rat) regulated by the members of 36 transcription factor families. The information is primarily provided in a table format but there are additionally static pictures generated of gene regulatory networks for each transcription factor family.

One of the most comprehensive cross-species compilation of transcription factor data and their DNA-binding sites is TRANSFAC (Matys et al., 2006). It contains *cis*-regulatory and *trans*-acting factor data on transcription factors, their experimentally proven binding sites, and regulated genes. All information is collected from primary experimental literature and no computational predicted data is included. Data of $\sim$ 1000 transcription factors are included in the current professional commercial version 2008.3 along with $\sim$ 33 000 regulated genes extracted from almost 19 000 references.

In contrast to the collection of information about regulatory factors of transcription given in literature the database JASPAR (Bryne et al., 2008) holds 138 curated, annotated, non-redundant, high-quality transcription factor binding site profiles as matrices for multicellular eukaryotes. JASPAR contains several sub-databases holding matrix models produced by different methods and for different purposes. Detailed information about each matrix model can be retrieved from JASPAR.

Despite the large number of databases containing biological data, a substan-

[3] In the context of transcriptional regulation, cis-acting elements or factors are considered to be DNA sequences, such as promoters or binding motifs.

[4] In the context of transcriptional regulation, trans-acting elements or factors are considered to be proteins that bind to cis-acting sequences to control gene expression.

tial amount of biological knowledge is still recorded in only free-text form and, hence, it is not readily available for computerized analysis. In order to extract meaningful information about gene regulation Jenssen et al. (2001) applied an automated analysis of titles and abstracts in over ten million MEDLINE records. They constructed a gene-to-gene network from co-occurrences of gene symbols or short gene names in the title or abstracts of the publications and showed that meaningful gene-gene co-regulations can be extracted.

All curated data in the databases described above are usually linked to the original publications and cross-referenced to multiple other databases to provide more information. Nevertheless, none of the databases can be considered to be complete, they rather complement each other. Although there is overlapping information, no database represent the full amount of information. Besides text-based queries in order to search and retrieve biological relevant information, the databases also implement a rich set of online analysis tools. For instance, YEASTRACT additionally provides a set of software tools for the identification of *de novo* binding sites from a given set of non-coding DNA sequences. RegulonDB offers sequence and gene expression analysis tools and a text mining engine. As another example, TRED database can be used to analyse internal or uploaded sequences. JASPAR includes algorithms for clustering selected position matrix models by similarity as well as an algorithm for the generation of random matrices by sampling from selected sets of existing models. Finally, the TRANSFAC Web interface provides several programs for transcription factor binding site searches.

Moreover, several databases give the possibility to download complete lists of transcriptional regulations or even the whole database content, e.g., RegulonDB and YEASTRACT. Other databases, such as JASPAR, provide access through Web Service application programming interfaces. All these information can help to set up computational models and gain comprehensive insights into the complex processes of transcription control.

## 1.3 Mathematical Background

### 1.3.1 Gene Regulatory Networks

The graphical representation of relationships or interactions between different biological entities is preeminently convenient to get directly a general idea and to gain a better global and local understanding of the whole intricate system. There are diverse types of graphs representing biological information, such as protein-protein interaction networks, metabolic networks, signaling pathways, phylogenetic networks, and gene regulatory networks, among others.

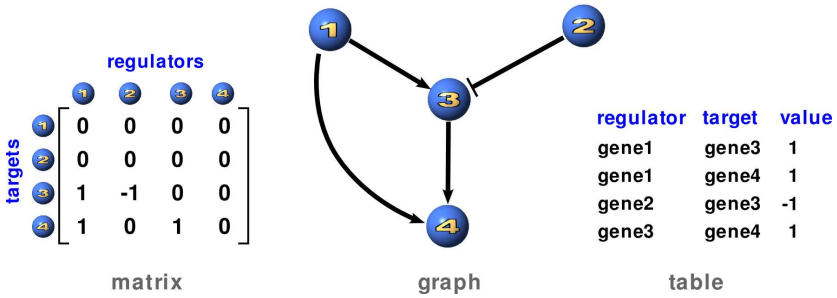



Figure 1.3 | **Representations of a gene regulatory network.** A directed graph is a graphical illustration of a gene regulatory network. Mathematically equivalently to this is the description of the graph by a ternary matrix or a list of regulations with associated values. Positive and negative effects of regulators on targets are distinguished in the graph by edges with arrows and bars, or in the matrix and table by $1$ and $-1$, respectively,

They vary in their represented entities and in the kind of relationships between them.

The collection of all direct regulatory effects of transcription factors on the expression of target genes can be compactly visualized by a directed graph defining a gene regulatory network (Figure 1.3). This can be mathematically represented by a graphical structure, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, composed of a set of $N$ nodes or vertices, $\mathcal{V} = \{V_1, \ldots, V_N\}$, and directed edges or links, $\mathcal{E} = \{(V_i, V_j) : V_i, V_j \in \mathcal{V}\}$, between two nodes[5]. In a gene regulatory network a node can represent a gene, the transcribed mRNA, and the coded protein simultaneously. A directed edge between two nodes indicates a regulatory interaction of a transcription factor (coded by the gene in the source node) on the transcription of a target gene (associated to the target node). The set of node indices[6] is denoted by $\mathcal{N} = \{1, \ldots, N\}$. In the following, components, i.e., genes, mRNAs, and proteins associated to the same node have the same index.

Similar network graphs are conceivable, where genes, proteins, and other instances of biological relevant objects are explicitly distinguished and depicted by different node types in the graph. Another alternative representation are bipartite graphs, where biological entities and reactions between them constitute different node types and edges connecting nodes of different types.

Besides its graphical representation, a network can be described either by an adjacency matrix, $\mathbf{W} = [w_{ij}]_{i,j \in \mathcal{N}}$ with $w_{ij} \in \mathcal{C}$, or a table or list of relations, $\mathcal{W} = \{(e, w) : e \in \mathcal{E} \wedge w \in \mathcal{C})\}$, where $\mathcal{C}$ is the set of possible values which can hold an edge (Figure 1.3). This set can be a binary set, $\mathcal{C} = \{0, 1\}$, a ternary set, $\mathcal{C} = \{-1, 0, 1\}$, or even all real values, $\mathcal{C} = \mathbb{R}$. Matrices with a binary

[5] $(V_i, V_j) \in \mathcal{E}$ is an edge from node $V_j$ to $V_i$.

[6] Nodes are often denoted by its index only, i.e, node $i \in \mathcal{N}$ references to $V_i \in \mathcal{V}$.

or ternary value set, $\mathcal{C}$, are sufficient and unique for the representation of the graphical structure whereas infinitely many matrices with real values can stand for the same graph. A ternary matrix entry, $w_{ij}$, is 1 or $-1$, if there is an edge, $e_{ij} = (V_i, V_j) \in \mathcal{E}$, from node $j$ to node $i$ where the value depends on the regulatory effect, i.e., 1 for a positive and $-1$ for a negative effect as depicted in the graph by arrow or bar ending edge, respectively[7]. Matrices with real values are also called weight matrices, where the weights assigned to the edges capture the degree or strength of the respective regulatory effects. In case of a directed graph, which is assumed usually for gene regulatory networks, the values $w_{ij}$ and $w_{ji}$ can be different. Furthermore, links from nodes pointing to itself are called self-loops.

A list or table of regulations, i.e., the list of regulator-target pairs together with an associated non-zero value can be used as a network representative as well. Biological networks usually have only a few links compared to the size of the network, i.e., they are sparsely connected (Thieffry et al., 1998), and, hence, the adjacency matrices have only a few non-zero values compared to the total number of entries, i.e., $|\mathcal{E}| \ll |\mathcal{V}|^2$. Therefore, a non-redundant list of regulations is much more convenient. Nevertheless, in mathematical formulae the matrix notation will be used.

Relevant for the interpretation of a gene regulatory network are also the set of regulators $\mathcal{R}_i = \mathcal{R}_{V_i} = \{V \in \mathcal{V} : (V_i, V) \in \mathcal{E}\}$ and targets $\mathcal{T}_i = \mathcal{T}_{V_i} = \{V \in \mathcal{V} : (V, V_i) \in \mathcal{E}\}$ that are specific for each gene $i \in \mathcal{N}$. The respective node indices are denoted by $\mathcal{N}_{R_i}$ for the regulators and $\mathcal{N}_{T_i}$ for the targets. The regulators are sometimes called parents and the targets children of the respective node in a more abstract sense.

### 1.3.2 Topological Characterization

The representation of gene regulatory networks as graphs make it possible to investigate the topology and function of these networks using several topological properties. These properties can be used to characterize and classify biological networks and to identify certain universal laws governed by networks or subnetworks with similar cellular functions (Aittokallio and Schwikowski, 2006). The topological measures give also insight into the evolution, stability, and dynamic responses of the system (Albert, 2005). Furthermore, using the knowledge about properties of well-studied networks can help to predict unexplored functional and dynamical properties of similar networks or indi-

[7]There is also a different definition of an adjacency matrix, where a non-zero entry $w_{ij}$ means a regulation from node $i$ on node $j$, i.e., this matrix is the transpose of the matrix $\mathbf{W}$ defined above. Nevertheless, the definition above is mathematically more convenient.

vidual components. A detailed review of statistical mechanics of complex network topology and dynamics compared to random networks is given by Albert and Barabási (2002). They discussed common features of biological and non-biological networks. Moreover, Barabási and Oltvai (2004) studied characteristics of biological networks and their functional organization and evolution. In the following, several measures will be described which are used in this thesis for the characterization of gene regulatory network structures.

The most elementary characteristic of a node $i$ in a directed graph are its in-degree, $k^i_{\text{in}}$, and out-degree, $k^i_{\text{out}}$. The in-degree is the number of incoming links to the respective node, i.e., it is the number of corresponding regulators, $k^i_{\text{in}} = |\mathcal{R}_i|$, and is equal to the number of all non-zero entries in the adjacency matrix row corresponding to this node. The out-degree is the number of outgoing links from the node, i.e., it is the number of corresponding targets, $k^i_{\text{out}} = |\mathcal{T}_i|$, and is equal to the number of all non-zero entries in the adjacency matrix column corresponding to this node. From the individual in- and out-degrees the corresponding distributions, denoted by $P_{\text{in}}(k)$ and $P_{\text{out}}(k)$, can be calculated by counting the number of nodes with $k = 1, 2, \ldots$ in-links and out-links, respectively, divided by the total number of nodes. The degree distributions give the probability that a selected node has exactly $k$ in-links and $k$ out-links, respectively, and is a global characteristic of a graph. The averaged in-degree, $\langle k_{\text{in}} \rangle = \sum_k k \cdot P_{\text{in}}(k)$, and out-degree, $\langle k_{\text{out}} \rangle = \sum_k k \cdot P_{\text{out}}(k)$, is another measure of a network. Since all out-going links have a target in the network, the average values are equal, i.e., $\langle k_{\text{in}} \rangle = \langle k_{\text{out}} \rangle = \langle k \rangle$.

A directed path, $s_{\text{AB}}$, between two (not necessary distinct) nodes A and B in a network graph is a sequence of nodes, $s_{\text{AB}} = [A, \ldots, B]$, starting with A such that following the edges between subsequent nodes by considering the direction node B can be reached from A. The length of a path is the number of nodes in the sequence minus one, i.e., $l_{\text{AB}} = |s_{\text{AB}}| - 1$. The shortest path between A and B is the sequence with the smallest number of nodes among all possible sequences. This is also called the distance between A and B. If there exists no such path between A and B the distance will be considered as zero. A circle in a network is a path where the start and end node is the same. The shortest path length distribution, $S(l)$, is the ratio of the number of shortest paths with length $l = 1, 2, \ldots$ among all shortest paths and is a further global network characteristic. The average shortest path length of a graph, $\langle l \rangle$, is the average over the shortest path lengths between all pairs of nodes and measures the typical distance or separation between two nodes in the graph. Furthermore, it gives an idea about the efficiency of the propagation of perturbations in a dynamical system.

The connectivity between the targets of a node $i$ is characterized by the clustering coefficient, $C_i$, introduced by Watts and Strogatz (1998). It is defined

as the number of links between the targets of node $i$ divided by the maximal number of links between them

$$C_i = \frac{\left|\{(V, \tilde{V}) \in \mathcal{E} : V \in \mathcal{T}_i \wedge \tilde{V} \in \mathcal{T}_V\}\right|}{k_i^{\text{out}}(k_i^{\text{out}} - 1)}, \tag{1.1}$$

assuming that $k_i^{\text{out}} > 1$. The value is between zero and one, i.e., $C_i \in [0, 1]$. For example, the node 1 of the graph in Figure 1.3 has a clustering coefficient of $C_1 = 1/2$ (one ($3 \rightarrow 4$) of two possible ($3 \rightarrow 4$ and $4 \rightarrow 3$) links between the target nodes 3 and 4). Nodes without any target are not considered for the clustering distribution, $C(k)$, which is defined as the averaged clustering coefficient of all nodes with $k = 1, 2, \ldots$ out-links. The average clustering coefficient, $\langle C \rangle$, over all nodes indicates the overall tendency of nodes to form cluster or groups. Sometimes it is termed short clustering coefficient (of the network).

The average values of the degrees, $\langle k_{\text{in}} \rangle$ and $\langle k_{\text{out}} \rangle$, the shortest path length $\langle l \rangle$, and clustering coefficient, $\langle C \rangle$, depend on the number of nodes and links in the network. In contrast, the distributions, $P(k)$ and $C(k)$, are independent of the network size and, hence, they capture generic features of the network.

With regard to the network measures mentioned above, networks can be classified into various network models, such as random networks, scale-free networks, small-world networks, and hierarchical networks (Barabási and Oltvai, 2004). They differ in one or more topological properties. It should be noted that networks having a similar topological measure can still differ strongly in other local or global properties. One of the most basic network class are random networks, since they represent unorganized systems and are used to emphasize certain features of other network types compared to random networks. Random networks are usually denoted as graphs in which the edges are distributed randomly. The probability for a directed edge between two arbitrary nodes is indicated by $p$ and represents the fraction of the number of present edges to all possible edges in a random graph. The properties of random networks were first defined and studied by Erdős and Rényi (1959).

Such networks are also called as Erdös-Renyi models. For large networks, i.e., $N \rightarrow \infty$, general properties can be determined. The average in- and out-degree is in this limit determined by $\langle k \rangle = |\mathcal{E}| / N = p(N-1)$, and the degree distributions can be approximated for large $N$ by a Poisson distribution (Albert and Barabási, 2002)

$$P(k) \simeq e^{-\langle k \rangle} \frac{\langle k \rangle^k}{k!}. \tag{1.2}$$

From this it follows that the majority of nodes have approximately the same degree, close to the average degree $\langle k \rangle$ of the network. The average shortest path

length of random graphs scales with the number of nodes in the network, i.e., $\langle l \rangle \sim \ln(N) / \ln(\langle k \rangle)$ (Chung and Lu, 2002). Finally, the probability, that two targets of an arbitrary node are connected is equal $p$ and, hence, the clustering coefficient of a random graph is $\langle C \rangle = p$ and is independent of the out-degree of a node.

Besides the above straightforward definition of random graphs, there are other types of random models. For instance, a random graph can be obtained by rearranging edges and preserving the degree distribution of a given graph. Randomized networks are generated by selecting randomly pairs of edges and switch their respective targets. Hence, nodes are connected without any preferences and evolutionary selection while having a certain degree-distribution. With an ensemble of such semi-random graphs, the statistical significance of certain features of the original network graph can be assessed (Artzy-Randrup et al., 2004).

One of the most important class of networks in biology is the class of scale-free networks, since many biological networks are supposed to be scale-free, including protein-protein interaction (Wagner and Fell, 2001), metabolic (Jeong et al., 2000), and transcriptional networks (Featherstone and Broadie, 2002). Also many non-biological networks share this property (Albert and Barabási, 2002). The degree distribution of scale-free networks follows a power law, i.e.,

$$P(k) \sim k^{-\gamma}, \tag{1.3}$$

where $\gamma$ is the degree exponent that can vary for the in- and out-degree in directed graphs. The degree distributions significantly deviates from a Poisson distribution with the main characteristic that most nodes have only a few links (in and out) and a few nodes, which are called hubs, are highly connected and hold the other nodes together. The average shortest path length compared to random networks is smaller and follows $\langle l \rangle \sim \ln \ln N$ (Barabási and Oltvai, 2004), but the clustering coefficient is independent of the degrees similar to random networks (Albert and Barabási, 2002). The concept of scale-free networks was first introduced and developed by Barabási and Albert (1999). Therefore, scale-free networks are sometimes also called Barabási-Albert model. Nevertheless, there are other scale-free networks which do not have the same properties as the Barabási-Albert network models.

A subclass of scale-free networks are so called small-world networks, which have besides a scale-free property in the degrees a small average shortest path length along with a large clustering coefficient compared to random networks. These types of networks were first introduced and studied by Watts and Strogatz (1998). Small-world networks are assumed to be more robust to perturbations than other network architectures. This could be a reason why in biological systems that are subject to damage by mutation or viral infection have a prevalence of small-world networks.

It is observed that many biological networks have not only the scale-free property, but also modules or clusters of highly interconnected group of nodes appearing in the graph (Hartwell et al., 1999). Such networks are called hierarchical networks (Ravasz et al., 2002) and have clustering coefficients significantly larger than those of random networks of equivalent size and degree distribution. This high clustering is an indicator for a small-world network. Additionally, the clustering coefficient, $\langle C \rangle$, is independent of the network size and the clustering distribution has a scale-free property, i.e., $C(k) \sim k^{-1}$. That means, there are many nodes with only a few out-links (scale-free property of the out-degree distribution), but they have high clustering coefficients, i.e., they belong to highly interconnected clusters and the few hub-nodes with low clustering coefficients, connect these clusters. In biological networks it is assumed that the cluster components work in conjunction to archive desired cellular functions.

The assumption that biological networks show certain properties, such as scale-free, is based on the identification and characterization of these properties in determined large-scale networks and extrapolation to properties of the whole, not completely covered system. This interpretation of global properties of the complete network structure should be made with caution and may fail. Stumpf et al. (2005) showed that under a certain sampling schema of subnetworks from a large scale-free network the subnetworks do not have necessarily as well a scale-free property in the degree. In turn, sampled networks may show scale-free property irrespective of the given complete network graph (Han et al., 2005). Furthermore, a recent study by Przulj et al. (2004) in interactome networks revealed that in protein-protein interaction networks the degree distribution do not have a scale-free behavior and can be better fitted by a geometric distribution. However, more important than the correct identification of the degree distribution is the functional characterization of biological networks with regard to clusters and small network modules.

Several global analyses of transcriptional regulatory networks in various organisms revealed that small network motifs, i.e., subnetworks showing certain characteristics, occur more often in gene regulatory networks than expected by chance (Shen-Orr et al., 2002, Milo et al., 2002, Lee et al., 2002, Mangan et al., 2003, Odom et al., 2004, Boyer et al., 2005). It is assumed that these recurring network motifs serve as basic building blocks of transcriptional networks and carry out specific functions (Alon, 2007). Significance of such motifs is typically assessed statistically by comparing the distribution of subgraphs in an observed network with that found in a particular random network serving as a null hypothesis. The correct choice of the random network is crucial for this statistical testing and is discussed, e.g., by Artzy-Randrup et al. (2004).

In Appendix A the structure and properties of the motifs identified by Lee

et al. (2002) are described in detail. These network motifs serve as basic building blocks for randomly generated networks as described in Chapter 2.

## 1.4 Reverse Engineering Methods

The basic assumption used by most reverse engineering algorithms is that causality of transcriptional regulation can be inferred from changes in mRNA expression profiles. The identification of the regulatory components of the gene expression is of major interest. Transcription factors bind to specific parts of DNA in the promoter region of a gene and, thus, effect the transcription of the gene. They can activate, enhance, or inhibit the transcription. Changes of abundances of transcription factors cause changes in the amount of transcripts of their target genes. This process is highly complex and interactions between transcription factors result in a more interwoven regulatory network. Besides this, transcriptional regulation can be affected as well on DNA and mRNA levels, e.g., by chemical and structural modifications of DNA or by blocking the translation of mRNAs by microRNAs (Ruvkun, 2001). Usually these additional regulation levels are neglected or included as hidden factors in the diverse reverse engineering models. Unfortunately, data on protein concentration measurements are currently not available in a sufficient quantity for incorporation in reverse engineering analysis. Therefore, gene expression profiles are most widely used as input for these algorithms.

Gardner and Faith (2005) classified reconstruction algorithms into two general strategies; *physical* approaches and *influence* approaches. Algorithms of the first group seek to identify interactions between transcription factors and DNA and reveal protein factors that physically control RNA synthesis. These methods, such as promoter binding analysis, as, e.g., performed by Lee et al. (2002), uses the genomic sequence information directly. The second strategy, the *influence* approach, aims to identify causal relationships between RNA transcripts by the examination of expression profiles. I will focus in the following on strategies based on the *influence* approach.

The reverse engineering strategies of the *influence* approach class can be further divided into model-driven and statistical approaches. The basic assumption of a model-driven approach is a model of gene regulation, represented by a structure $\mathcal{M} = (\mathcal{G}, \mathcal{X}, \mathcal{F}, \mathcal{Q})$ with a graph, $\mathcal{G}$, a set of variables, $\mathcal{X}$, a set of functions, $\mathcal{F}$, and a set of parameters, $\mathcal{Q}$, of these functions. The structure of a graph, $\mathcal{G}$, is described in the previous Section 1.3.1. Each node $i$ in the graph $\mathcal{G}$ is a variable $X_i \in \mathcal{X}$ assigned which has a value $x_i \in \mathbb{R}$. The vector of all values is denoted by $\mathbf{x} = (x_1, \ldots, x_N)$. Additionally, variables associated to the regulators (or parents) of node $i$ are collected in the set $\mathcal{X}_{\mathcal{R}_i}$. The value

vector of the regulators of node $i$ is depicted by $\mathbf{r}_i = (r_i^1, \ldots, r_i^{k_{\text{in}}^i})$ with values $r_i^\xi = x_{j_i^\xi}$, $\xi = 1, \ldots, k_{\text{in}}^i$, and $j_i^\xi$ the $\xi$th element of the index vector $\mathbf{j}_i$ of regulator indices[8] in $\mathcal{R}_i$. This vector defines the input state of node $i$. A function $f_i \in \mathcal{F}$ with $f_i : \mathbb{R}^{k_{\text{in}}^i} \to \mathbb{R}$ is assigned to each target node, representing a regulation of dependence relation between the node and its regulators.

A computational model for reconstruction purposes is an approximation of a real biological system and should reflect features of the real system. Nevertheless, the model has to be simplified to reduce the complexity and the number of parameters which are fitted to given data. A balance between these contradictory views has to be accommodated. Associated to a model-driven approach are learning strategies to assess one or multiple sets of model parameters which give the best results according to an approach specific validation measure with regard to the given data. The learning strategies may differ within a model type and, even more, across the diverse model types. Often the simplest solution and most parsimonious model is favored by the learning algorithm despite the possibility that a more complex model is biologically more plausible. Without any prior knowledge such a decision is impossible.

In contrast to model-driven approaches, statistical reverse engineering approaches, such as associated networks and graphical Gaussian models (Basso et al., 2005, Schäfer and Strimmer, 2005a) lack a gene regulation model whose parameters have to be fitted. The data is rather analyzed directly by means of statistical strategies to reveal direct, not always directed, regulatory relationships between transcription factors and genes.

Reverse engineering methods have diverse properties and they differ in their strategy to tackle the difficulties in reconstruction of the underlying gene regulatory network. There are static or dynamic, continuous or discrete, linear or nonlinear, deterministic or stochastic models. They can differ in the information they provide and, thus, have to be interpreted differently. Some methods result in correlation measures of genes (relevance networks), some calculate conditional independencies (Bayesian approaches), and others infer regulation strengths (linear models). These results can be visualized as directed or undirected graphs representing the inferred gene regulatory networks. For that, a discretization of the results given as real value adjacency matrix is necessary for some methods. Each concept has certain advantages and disadvantages which are revealed by various comparison studies, such as Wessels et al. (2001), Werhli et al. (2006), Bansal et al. (2007), Camacho et al. (2007), and Soranzo et al. (2007).

[8]The index vector $\mathbf{j}_i = (j_i^1, \ldots, j_i^{k_{\text{in}}^i})$ of all regulator indices of node $i$ is ordered, i.e., $j_i^\xi \in \mathcal{R}_i$ $\forall \xi$ and $j_i^p < j_i^q$ $p, q = 1, \ldots, k_{\text{in}}^i \wedge p < q$.

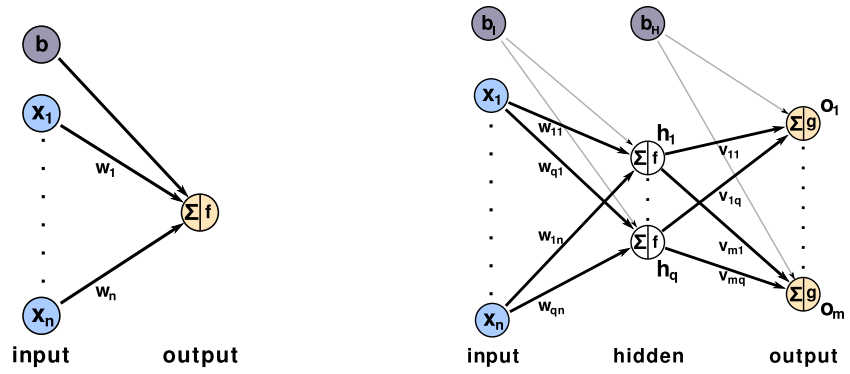
Several model-driven reverse engineering methods were proposed in recent years which are based on diverse mathematical models, such as Boolean networks (Liang et al., 1998), linear models (D'haeseleer et al., 1999), differential equations (Chen et al., 1999), static Bayesian networks (Friedman et al., 2000), state space models (Rangel et al., 2004, Beal et al., 2005), and dynamic Bayesian networks (Friedman et al., 1998, Yu et al., 2004, Werhli et al., 2006). In Section 4.1 I will discuss basic features, differences, and applications following model-driven approaches as well as statistical-approaches, such as relevance networks (Basso et al., 2005) and graphical Gaussian models (Schäfer and Strimmer, 2005b). A large comparative study comprising various methods mentioned above, including a method based on neural networks and developed by me was conducted in the course of my thesis and will be discussed in Chapter 4 on page 81.

Besides the models mentioned above, there are some other approaches which are different in principle, like Petri Nets (Petri, 1962, Matsuno et al., 2000, Marwan et al., 2005) and decision trees (Soinov et al., 2003). Particularly with ordinary differential equations (ODEs) a very detailed description of a (dynamical) gene regulation system is possible. It is a widely used technique for realistic forward modeling of biological systems but also applied to reverse engineering (Chen et al., 1999). Several learning techniques for differential equation systems are used, e.g., genetic algorithm (GA) (Wahde and Hertz, 2000), single value decomposition (SVD) (Yeung et al., 2002), simulated annealing (Chen et al., 2001), and algebraic approaches (Laubenbacher and Stigler, 2004).

### 1.4.1 Neural Networks

Biological neural networks are composed of real biological neurons that process and transmit information by electrochemical signals to other physically connected neurons. Inspired by biological neural networks artificial neural networks were developed. The mathematical concept of such networks, in the following denoted as neural networks, has its origin in the desire to model the cognitive processes of recognition and learning. Neural networks consist of a multiplicity of simple processing units, called neurons, which exchange information between each other simultaneously.

A simplistic model of a neuron in the central nervous system was first developed by McCulloch and Pitts (1943) using electrical circuits. Such a neuron, as illustrated in Figure 1.4a, has several weighted binary input signals which are added up and processes an output signal with an activation or transfer function $f$. The authors proposed a step like activation function, i.e., if the sum exceeds a pre-defined threshold depicted by $b$, the neuron is activated. A continuous model of a neuron was proposed by Rosenblatt (1958). It is called perceptron




(a) Single-Layer Perceptron (b) Multi-Layer Perceptron

Figure 1.4 | **Different perceptron models. a** | Single-Layer perceptron. It is a simple model of a neuron and has an input and an output layer. The effects of all input neurons, represented by $x_i w_i \ \forall i$, are added up. $b$ depicts an additional bias. The total signal is processed by function $f$. **b** | Multi-Layer perceptron. It has an input and an output layer and several hidden layers (only one is shown here). Signals in the hidden and output layer are processed of each node similar to a neuron in a single-layer perceptron with different transfer functions.

and has the same structure as the neuron proposed by McCulloch and Pitts except that the activation function is a continuous sigmoidal function. Further developments leads to a multi-layer percepton, shown in Figure 1.4b.

The main feature of neural networks is the capacity to learn from a given set of examples. That means in principle, providing an input vector to the adapted system a desired output is processed. Learning from data is achieved by altering the weights of the connections in the network or by including new links or removing existing links. It is remarkable that the information is not stored at single processing units but all over the network in the connection weights. Choosing a sufficient learning strategy for a neural network is of decisive importance for its application. Despite the simplicity of the local structure of a neuron, the complete system has complex properties. This leads to numerous application fields. For instance, neural networks are successfully applied to gesture, speech, and handwriting recognition, image analysis, classification problems, function approximation, and data processing including filtering and clustering.

Neural networks are also used for modeling gene regulation and gene expression (Vohradsky, 2001a). Their artificial model aims to reproduce the dynamic of biological systems without any hidden layers. Different expression patterns, such as relaxation and oscillation can be modeled with this approach. A concrete model of gene regulation was proposed by Vohradsky (2001b) for the $\lambda$ bacteriophage lysis/lysogeny decision circuit. The model includes several genes which are known to be involved in the switch between the lysic and

lysogenic pathway. They showed that their model is in agreement with experimental observations and demonstrated that neural networks are suitable to model biological phenomenons.

Since neural networks have several valuable features I have used this concept for the development of a reverse engineering method to study biological gene regulatory networks in Chapter 3. With neural networks simple but powerful models of gene regulation can be established, represented by a set of parameters and depicted graphically by a network graph. Neural networks give the possibility to model multigenic regulation including activation, inhibition, and self-regulation. Furthermore, several dynamical features of gene regulatory systems can be reproduced. The complete model is mathematically described by a system of ordinary differential equations, which can be treated by various well developed mathematical approaches.

## 1.5 Objectives and Outline

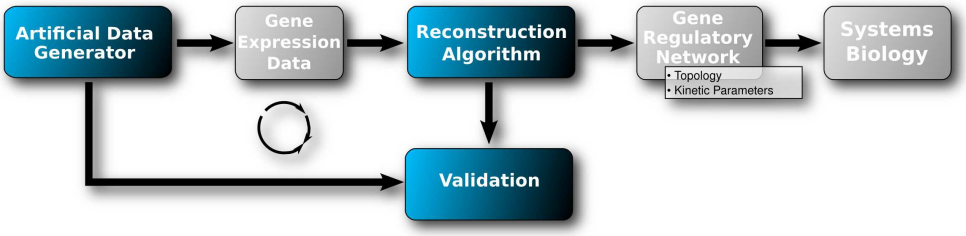



Figure 1.5 | **Framework for modeling gene regulation.** In the course of my thesis I have addressed the two fundamental approaches to gene expression analysis, forward modeling and reverse engineering. I have developed an artificial data generator, a method for reverse engineering, and a validation framework for assessing method performances (blue boxes)

The computational analysis of gene regulatory networks in my thesis addresses several aspects highlighted in Figure 1.5. A systematic evaluation and validation of reverse engineering methods is facilitated through artificial benchmark data. In the course of my thesis, I have developed a data generator for artificial gene regulatory networks called Gene Network Generator (GeNGe) (Hache et al., 2009b). GeNGe has been designed for forward modeling, for example to investigate performances of reverse engineering methods under well-defined conditions, to analyse general features of regulatory networks using topological and dynamical parameters, or to predict network properties with respect to perturbations as theoretical counterparts of experiments. In Chapter 2 I introduce GeNGe, describe its functionalities, and highlight unique features that are not provided by any other software applications currently available.

For the dynamical description of the gene regulatory models I have derived a kinetic from a general description of gene regulation by Schilstra and Nehaniv (2008). This new kinetic is preeminently convenient for the computational set up of complex gene regulatory models. This is demonstrated by means of several examples.

The second major focus of my thesis is the methodological development for reverse engineering. I have developed a method called GNRevealer for the analysis of time course data based on neural networks together with the Backpropagation through time learning algorithm (Hache et al., 2007). This is the first time that this approach has been implemented for the analysis of gene expression regulation. In Chapter 3 I go into mathematical details of GNRevealer and the learning algorithm, describe important features of the workflow, show its ability to cope with high dimensional complex datasets, and highlight learning properties with respect to network motifs.

In Chapter 4, I discuss the validation of reverse engineering algorithms. A large comparative study was conducted, comprising several reverse engineering methods, including relevance networks, Bayesian networks, and the GNRevealer. The approach consists of three steps: (i) the generation of defined benchmark data by GeNGe; (ii) the analysis of these data with the different methods, and (iii) the assessment of algorithmic performances by statistical analyses. Performances are judged by network size and noise levels. The results of the comparative study presented in this chapter highlight the neural network approach GNRevealer as best performing method among those under study (Hache et al., 2009a).

In summary, my thesis presents a comprehensive theoretic analysis of gene expression regulation on different levels of investigation. It resulted in two fully-operational computational tools, GeNGe and GNRevealer, that cover the two main aspects of gene regulatory network analysis.

# CHAPTER 2

# Forward Modeling: Generation of Artificial Benchmark Data with GeNGe

Modeling of biological systems is indispensable in various fields in computational biology. For instance, there is an increasing interest in computational models of gene regulatory systems for the development of reverse engineering methods. Availability of experimental data is still the major bottleneck for benchmarking such methods. Thus, generating simulated data derived from theoretical considerations is still the method of choice for a systematic evaluation and validation of such methods. For realistic performance assessments adequate mathematical descriptions of gene regulatory systems comprising plausible network structures and appropriate kinetics are of fundamental importance. The modeling and simulation of such models should have to be carried out automatically to a large extent. However, there is a lack of tools which are targeted on this problem domain.

In the course of my thesis I have developed a Web application, called Gene Network Generator (GeNGe), for the analysis of gene regulatory networks with a modeling and simulation approach (Hache et al., 2009b). A new network generation algorithm is presented to provided large numbers of networks. A topological characterization of such networks revealed several biological features. These features are compared to those of other network types

and a gene regulatory network from TRANSFAC. For the dynamical description of gene regulatory models I have derived a transcription function which allows modeling of several dynamical features of biological systems demonstrated with several examples.

## 2.1 Modeling of Transcriptional Regulation

A computational model of a biological system is always a simplified image of the reality due to the high complexity of biological systems, huge amount of involved components, lack of detailed knowledge about these components and their intricate interactions, and computational limitations. Biological models are usually described by reduced and computationally suitable equation systems. Despite of these simplifications, computational modeling is considered as a valuable approach for gaining a better understanding of processes which are experimentally difficult to investigate.

Gene expression is a complex process regulated at many levels as described in Section 1.1.2. Therefore, the development of a comprehensive mathematical model is a complicated task. Various formalisms have been employed for the mathematical description of genetic regulatory systems (Hasty et al., 2001, de Jong, 2002). There are different abstraction levels of such formalisms due to limited information about involved components, their kinetics, and the modeling purpose. For instance, Boolean networks can be used as a first approximation of regulatory events, where the state of a gene is either on or off and interactions are Boolean functions. General properties can be derived from such an approach. However, ordinary differential equations (ODEs) are the most widespread formalisms to describe quantitatively dynamic gene regulation. Highly detailed models can be constructed. However, genes and proteins serving as transcription factors are identified as the key players in transcriptional regulation. At least these components have to be comprised by a simple computational model of gene regulation. In a refined model intermediate levels of interactions and other entities, such as mRNAs, are incorporated.

### 2.1.1 Simplified Gene Regulation Model

In the following I describe a simplified model of gene regulation which is computationally tractable. It is preeminently suitable for the automatic set-up of large gene regulatory models and is adjustable by means of a few parameters resulting in various behaviours. This simplified model of gene regulation is given by a mathematical model that covers the transcriptional as well as the translational layer shown in Figure 2.1. These layers include instances for

mRNAs and proteins as well as for a polymerase and a ribosome. Furthermore, an RNase and a proteasome instance catalyze the mRNA and protein degradation, respectively. The kinetics of the concentration of an mRNA and a protein, associated to a certain gene, are described by rate equations

$$\frac{\mathrm{d}[\mathrm{mRNA}]}{\mathrm{d}t} = k_1[\mathrm{Polymerase}]\phi_n(c_{t_1},\ldots,c_{t_n}) - k_2[\mathrm{RNase}]\delta([\mathrm{mRNA}]), \tag{2.1a}$$

$$\frac{\mathrm{d}[\mathrm{Protein}]}{\mathrm{d}t} = k_3[\mathrm{Ribosome}][\mathrm{mRNA}] - k_4[\mathrm{Proteasome}]\delta([\mathrm{Protein}]), \tag{2.1b}$$

including transcription and degradation of the mRNA (Eq. (2.1a)) and translation and degradation of the protein (Eq. (2.1b)). The non-linear function $\phi_n$ describes the transcriptional regulation depending on concentrations $\mathbf{c} = (c_{t_1},\ldots,c_{t_n})$ of the transcription factors $t_i \in \mathcal{N}_{R_i}$. The kinetic parameters $k_1$ and $k_3$ are the transcriptional rate constant of the mRNA and the translational rate constant of the corresponding protein, respectively. $k_2$ and $k_4$ are the respective degradation rates. The parameter $k_1$, reflects the RNA polymerase translocation rate. The maximal transcription rate or maximal initiation rate is the product of the rate constant, $k_1$, the polymerase concentration, and the maximal value of $\phi_n$. This maximal rate is the concentration of mRNA molecules synthesized per time unit at optimal circumstances. Furthermore, the parameter $k_3$ represents the efficiency of the translation by the ribosome. The maximal translation (initiation) rate is the product of $k_3$ and the ribosome concentration. It is the optimal rate for translating the mRNA code to an amino acid sequence per time and per mRNA concentration. This rate is largely independent of the coding sequence and is similar for all proteins in a given cell type at a given condition (de Leon and Davidson, 2009).

In Eq. (2.1b) translation is described by the first term on the right hand side with a linear kinetic. Similarly, degradation is usually modeled with a linear kinetic ($\delta(x) = x$). However, it is observed, that protein degradation can also be described by a Michaelis-Menten kinetic ($\delta(x) = x/(K_M + x)$) (Grilly et al., 2007). For that, an additional parameter, $K_M$, specific for each mRNA and protein is introduced for the Michaelis-Menten kinetic, representing the substrate concentration. This defines the mRNA or protein concentration, at which the reaction rate reaches half of its maximum value. Other non-linear degradation kinetics are conceivable especially for multimeric proteins, that results in a concentration dependent degradation rate (Buchler et al., 2005). This effect is called cooperative stability and can be important for robust operation of genetic circuits. These specific effects are not considered in the simplified model.

In case of a linear degradation rate and a constant enzyme concentration the

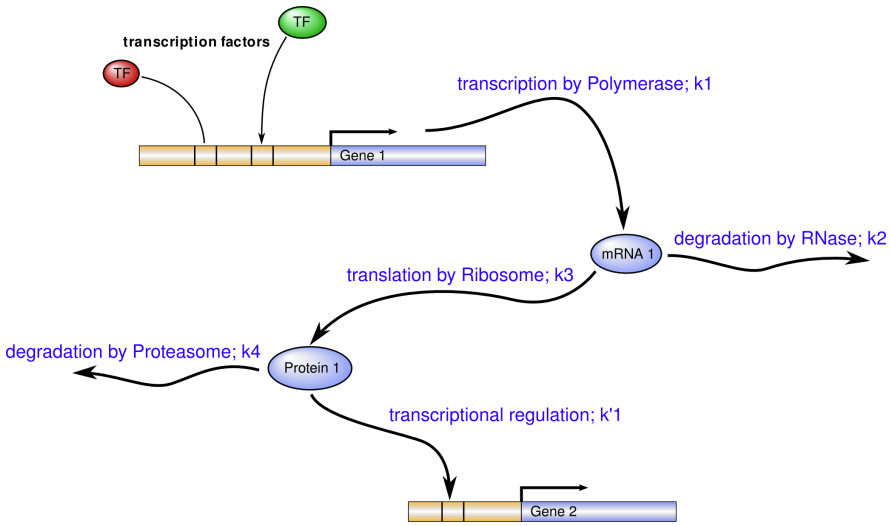



Figure 2.1 | **Simplified model of gene regulation.** The model comprises a transcriptional and a translational layer. Genes are transcribed to mRNAs and translated into proteins catalyzed by certain enzymes. Some proteins serve as regulators for the transcription of genes. Furthermore, mRNAs and proteins are subject to degradation. Processes described by mathematical equations in the simplified model are depicted in blue.

half-life, $\tau_{1/2}$, of an mRNA and a protein can be expressed by

$$\tau_{1/2}^{\text{mRNA}} = \frac{\ln 2}{k_2[\text{RNase}]} \quad \text{and} \quad \tau_{1/2}^{\text{protein}} = \frac{\ln 2}{k_4[\text{Protesome}]}. \tag{2.2}$$

If the degradation is described by a Michaelis-Menten kinetic or another non-linear kinetic, the half-life cannot be expressed independently of the initial concentrations as in the linear case.

The function $\phi_n(c_{t_1}, \ldots, c_{t_n})$ in Eq. (2.1a) is the transcription (rate) function, regulating the efficiency of transcription of the gene. Its value is always equal or larger than zero and depends on $n$ concentrations $\mathbf{c} = (c_{t_1}, \ldots, c_{t_n})$ of all proteins $t_i \in \mathcal{N}_{R_i}$ acting as transcription factors. $\mathcal{N}_{R_i}$ is the set of indices of all transcription factors of the gene. The function is described briefly below and in more details in Appendix B on page 115.

Note that all parameters, $k_1, \ldots, k_4$, the number of transcription factors, $n$, the set of transcription factor indices, $\mathcal{N}_{R_i}$, and the transcription function, $\phi_n$, are gene specific and can vary between genes. However, an additional index is omitted throughout this thesis as long as it does not lead to confusions.

A gene regulatory system of $N$ genes, whereas corresponding mRNA and protein dynamics are described by the rate equations Eq. (2.1a) and Eq. (2.1b), is a system of $2N$ coupled, non-linear, first-order differential equations which has to be solved with regard to the mRNA and protein concentrations. For

that, it is usually assumed that the parameters and the enzyme concentrations are constant over time.

I have adapted the *bio-logic* proposed by Schilstra and Nehaniv (2008) for the description of transcription kinetics. It is a thermodynamic approach based on the assumption that gene regulation is controlled completely by the equilibrium binding of proteins to DNA, which form DNA-TF complexes. Other regulatory events, such as chromatin modification and polymerase phosphorylation are not taken into account. Furthermore, the theory assumes, that the number of activated transcription factors which are present in the cell is much larger than the number of binding sites where they can bind. Gene expression is regulated by specific protein-DNA interactions, i.e., interactions between transcription factors and a cis-regulatory domain of the gene. They affect the initiation rate of the transcription of the target gene to a certain extent.

The initiation rate expressed by the transcription function, $\phi_n$, is composed of the fractional saturation functions of each possible complex $\mathrm{DNA} \circ \mathrm{TF}_1 \circ \ldots$ of DNA and zero, one, or more transcription factors. The fractional saturation of such a particular complex is considered as the fraction of time that this complex exists. It is primarily dependent on the concentrations of the transcription factors.

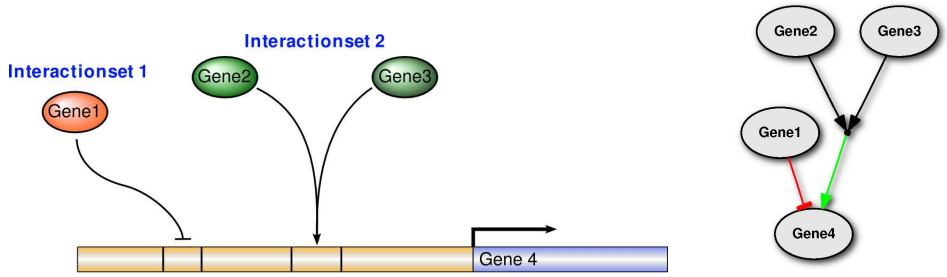
The transcription function of a gene with $n$ transcription factors,

$$\phi_n \propto \left(w_0\varphi_0 + w_1\varphi_1 + \cdots + w_{2^n-1}\varphi_{2^n-1}\right), \tag{2.3}$$

is proportional to the weighted sum of rates of the individual complex contributions. A contribution of a complex is denoted by its fractional saturation function $\varphi$. A weight, $w$, of a complex[1] is also called modulation coefficient. Its value is independent of the concentrations of the contributed transcription factors and the fraction of time the associated complex exists, but it is determined by the nature of the complex. The weights are larger or equal to zero, where larger values indicate a stronger contribution to the initiation rate of the gene's transcription. An additional cooperative factor as considered in Appendix B for each term, which reflects dependencies in the binding of transcription factors in a complex, is here omitted.

For the description of complex dynamics I have introduced the concept of jointly binding transcription factors. It is known, that transcription factors can bind together in a highly dependent way at a regulatory region and have a cooperative effect on the target transcription. They define a joint interaction set $\mathcal{I}$. Such a set regulates the transcription only if all transcription factors of this set bind to DNA jointly, otherwise, they do not have any effect. To form an

[1]The weights are here numbered serially for simplicity. In Appendix B there is a different and more general notation.




(a) Illustration (b) Graph

Figure 2.2 | **Illustration of joint interaction sets. a** | Interaction sets bind to the regulatory region of the target gene and, thus, regulate the transcription of this gene. It is assumed, that each interaction set binds independently from each other. **b** | Graph of a four node example. The nodes represent the gene and the corresponding protein, which can act as a transcription factor. A green edge indicates a positive regulatory effect of the transcription factor on the transcription of the connected gene (activation) and a red edge ended with a bar indicates a negative effect (inhibition). Some transcription factors (*Gene2* and *Gene3*) have a regulatory effect only, if they bind jointly. These are combined to interaction sets, indicated in the graph by black edges to a joint node. The type of regulation of an interaction set, i.e., activation or inhibition, is represented by a green or red edge, respectively, from the joint node to the target gene. Single transcription factors (*Gene1*) are considered as interaction sets with one element.

interaction complex of $m$ transcription factors and DNA, there is a $m+1$-body-collision required in the nucleus.

If there are joint interaction sets present as regulatory units for a gene, the transcription function, Eq. (2.3), has to be changed. All terms representing complexes that contain one or more transcription factors of an interaction set are neglected unless all members of a set are present. Only those terms remain which represent complete interaction sets.

A gene can have several joint interaction sets, $\mathcal{I}_1, \ldots, \mathcal{I}_\nu$, with $\nu \leq n$ as regulatory units with the assumption that transcription factors acting alone define also a joint interaction set. Each set, $\mathcal{I}_i$, contains one or more transcription factors[2] and each transcription factor is in exactly one set, but can occur more than one time in this set. Thus, all sets are disjoint and the conjunction of all sets is equal to the set of all transcription factor indices. Therefore,

$$\mathcal{I}_i \cap \mathcal{I}_j = \emptyset \;\; \forall i,j \in \{1,\ldots,\nu\} \wedge i \neq j \quad \text{and} \quad \mathcal{I}_1 \cup \cdots \cup \mathcal{I}_\nu = \mathcal{T}. \tag{2.4}$$

I have derived a transcription function from the general description developed by Schilstra and Nehaniv (2008). This new kinetic includes the assumption that each joint interaction set acts independently of each other. Furthermore, interaction sets have an individual regulation strength representing the

[2]More precisely, $\mathcal{I}_i$ is a set of indices of associated transcription factors.

regulatory effect on the transcription, i.e., the extent of activation or inhibition. In such a single set, a transcription factor can occur more than once to model homomers. With these constraints the following transcription function is obtained[3]

$$\phi_n(c_{t_1},\ldots,c_{t_n}) = \prod_{i=1}^{\nu}\left[1 + (2^{a_i}-1)\frac{\tilde{x}_i}{1+\tilde{x}_i}\right], \tag{2.5}$$

where $n$ is the number of transcription factors, $t_i \in \mathcal{N}_{R_i}$, which combine to $\nu$ interaction sets. The set of regulation strengths of the interaction sets are collected in $\mathbf{A} = (a_1,\ldots,a_\nu)$ with $a_i \in \mathbb{R}$. $\tilde{x}_i$ denotes the product of all normalized transcription factor concentrations, $x_j$, in interaction set $i$,

$$\tilde{x}_i = \prod_{j\in\mathcal{I}_i} x_j \qquad \text{with } x_j = \frac{c_j}{K_j}, \tag{2.6}$$

with $c_j \in \mathbf{c}$. The definition of the normalization constants, $\mathbf{K} = (K_1,\ldots,K_\nu)$, differ for single transcription factors and transcription factors in a joint interaction set. For single transcription factors the normalization constants are equal to the dissociation constants of the transcription factor-DNA complex. On the contrary, for transcription factors in joint interaction sets the normalization constants are $K_i^{1/\dim(\mathcal{I}_i)}$ with $K_i$ as the dissociation constant of the whole interaction set-DNA complex.

The transcription of a gene with no interaction sets other than single transcription factors and a regulation strength $a_i = 1$ for all activators $A = \{i \in \{1,\ldots,\nu\} : \mathbf{A}_i = a_i > 0]\}$ and $a_i = -\infty$ for all inhibitors[4] $I = \{i \in \{1,\ldots,\nu\} : \mathbf{A}_i = a_i < 0]\}$ is described by the transcription function

$$\phi_n(c_{t_1},\ldots,c_{t_n}) = \prod_{i\in A}\left[1 + \frac{\tilde{x}_i}{1+\tilde{x}_i}\right]\prod_{i\in I}\left[1 - \frac{\tilde{x}_i}{1+\tilde{x}_i}\right], \tag{2.7}$$

where the first product is over all activators and the latter over all inhibitors. This kinetic law term with only single transcription factors is already described by Mendes et al. (2003) and is used often for modeling transcriptional regulation.

A slightly different transcription function compared to Eq. (2.5) can be derived under different constraints. For the following transcription function it

[3] See Appendix B on page 115 for a detailed derivation of the function and similar transcription functions.

[4] $a_i = -\infty$ is a symbolic representation of $a_i \to -\infty$ and $\lim_{a_i\to-\infty} 2^{a_i} = 0$.

Table 2.1 | **Response characteristics.** Depending on the concentration values of the transcription factors the transcription rate of the target gene shows different response characteristics for the different models shown in Figure 2.3 and Figure 2.4. A low concentration means values $\sim 1\,\mathrm{nM}$ and high values $\sim 500\,\mathrm{nM}$. Certain logic functions are implemented: an OR logic (Gene1), AND logic (Gene2), NAND logic (Gene3), and the composed logic function NOT TF1 AND TF2 (Gene4), and NOT TF1 AND NOT TF1 (Gene5).

| TF1 | TF2 | Gene1 | Gene2 | Gene3 | Gene4 | Gene5 |
|---|---|---|---|---|---|---|
| low | low | off | off | on | off | on |
| high | low | on | off | on | off | off |
| low | high | on | off | on | on | off |
| high | high | on | on | off | off | off |

is assumed that there is no basal expression of the target gene, i.e., the contribution of unbounded DNA is neglected in Eq. (2.3), i.e., $w_0 = 0$. Then the transcription function is given by

$$\phi_n(c_{t_1}, \ldots, c_{t_n}) = \prod_{i=1}^{\nu} \left[ 1 + (2^{a_i} - 1) \frac{\tilde{x}_i}{1 + \tilde{x}_i} \right] - \prod_{i=1}^{\nu} \frac{1}{1 + \tilde{x}_i}, \tag{2.8}$$

where the second term on the right hand side realizes the additional constraint.

Although the protein concentrations are continuous, it is possible to model certain binary response characteristics with the derived transcription kinetics realized by regulation strengths $\mathbf{A}$ and joint interaction sets. An overview of some possible response characteristics is given in Table 2.1 and the respective transcription factor curves are shown in Figure 2.3 and Figure 2.4. The activation input patterns are composed of low transcription factor values ($\sim 1\,\mathrm{nM}$) and high values ($\sim 500\,\mathrm{nM}$). The resulting transcription factor function has different output states, where larger values correspond to an activation of the target gene (*on*-state) and lower levels corresponds to an expression at a basal level or even a complete turn-off (*off*-state). The state is indicated by colors in the figures; red for *on*- and green for *off*-state.

Several logic functions are implemented by using Eq. (2.5) or Eq. (2.8). Certain combinations of regulation strengths, $\mathbf{A}$, and joint interaction sets as a system of two transcription factors and one regulated gene realizes various binary logics, such as AND, OR, and NAND. The corresponding transcription function curves are plotted in Figure 2.3 and Figure 2.4. For instance, in the OR-logic (first case in Table 2.1), *Gene1* is transcribed, if either TF1 or TF2 is present in a sufficient concentration. In that case they can bind to DNA and activate transcription independently from each other. The transcription rate is highest, if both factors are bound. In contrast, with an AND logic (case 2),

the transcription factors act highly cooperative as a joint interaction set. Each factor cannot bind alone to its site, but activate *Gene2* conjointly. The inverse logic, i.e., NAND logic is realized similarly, but assumes an inhibitory joint interaction set. Two further examples are given that implement composed logic functions. *Gene4* is turned on by TF2 only if TF1 is not present, since TF1 is in case 4 a very strong inhibitor, i.e., $a_i \ll 0$ and has a stronger effect than activator TF2. Changing the activator TF2 into an inhibitor like TF1 results in a different response logic NOT TF1 AND NOT TF2 (case 5).

A faster or sharper change between *on* and *off* can be achieved by assuming exclusively joint interaction set binding. The models shown in Figure 2.3 are reimplemented and visualized in Figure 2.4 under the assumption that all transcription factors act as homodimers[5]. The models represent the same logic and the curves show the same response characteristics but with a faster switch between different output states. Furthermore, different fold changes between *on* and *off* state-values can be obtained by changing the regulation strengths $\mathbf{A}$.

I used the kinetics described above as the basis for the modeling of gene regulatory systems within the GeNGe system. GeNGe will be introduced below in the Section 2.2. Joint interaction sets and regulation strengths, $\mathbf{A}$, which are introduced above, capture different effects of transcription factors in more details than it is possible in similar functions used in this context. All parameters are adjustable by the user within GeNGe to provide rich functionalities to model various dynamics.

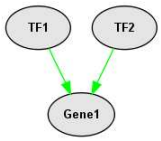
### 2.1.2 State-Of-The-Art in Forward Modeling Gene Regulatory Networks

There are some tools previously published that provide forward gene regulatory network modeling approaches, such as SynTReN (den Bulcke et al., 2006), RMBNToolbox (Aho et al., 2007), RENCO (Roy et al., 2008), and SynBioSS (Hill et al., 2008). All of them have at least one similar focus, namely the generation of synthetic datasets, which can be used, e.g., for a systematic validation of network inference methods. Nevertheless, the functionalities vary across the tools substantially, mainly regarding their various *in silico* experiments and analysis features. In the following I review some of the tools since they express the state-of-the-art in forward modeling tools for gene regulatory networks and were the benchmark for the development of GeNGe.

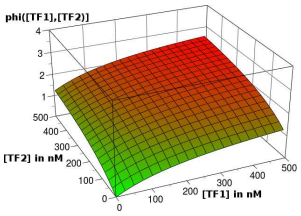
SynTReN (Synthetic Transcriptional Regulatory Networks) was developed by den Bulcke et al. (2006). It is implemented in Java and is running on a local

[5] A homodimer is represented as a node with two black edges pointing to a joint black node, i.e., it is an interaction set with two entities representing the same transcription factor.

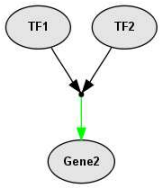
### Case 1: OR logic



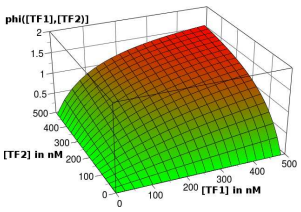

Eq. (2.8) with
$a_1 = 1$
$a_2 = 1$



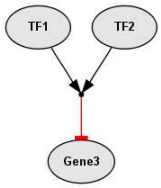

### Case 2: AND logic



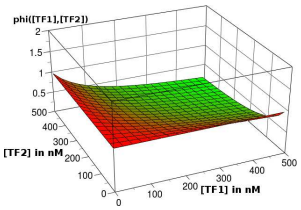

Eq. (2.8) with
$a_1 = 1$



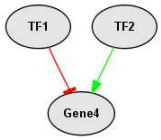

### Case 3: NAND logic



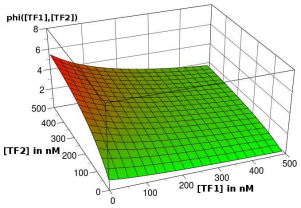

Eq. (2.5) with
$a_1 = 1$



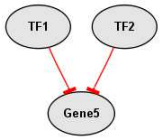

### Case 4: NOT TF1 AND TF2 logic




Eq. (2.5) with
$a_1 = -\infty$
$a_2 = 3$




### Case 5: NOT TF1 AND NOT TF2 logic




Eq. (2.5) with
$a_1 = -\infty$
$a_2 = -\infty$

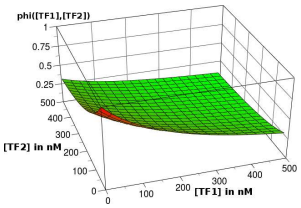
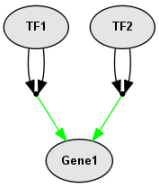
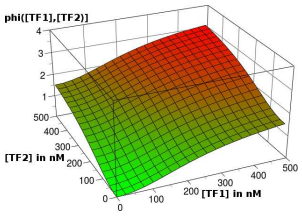



Figure 2.3 | **Response characteristic for various transcription models.** In the left column a three-node network representation of the corresponding logic is shown (see Figure 2.2 for graph explanations). The kinetic parameters for the given transcription function, are given in the middle column (normalization constants $K_i \in \mathbf{K}$ are set $K_i = 250$). The corresponding response characteristic as the function of transcription factor concentrations in a reasonable range is shown in the right column. Green and red indicate low and high function values, respectively.

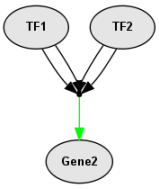
### Case 1: OR logic



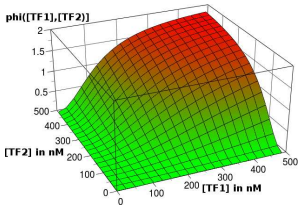

Eq. (2.8) with
$a_1 = 1$
$a_2 = 1$



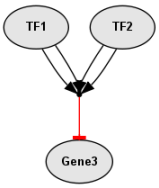

### Case 2: AND logic



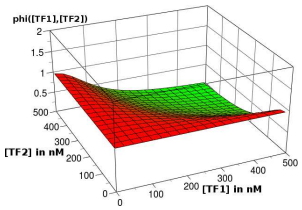

Eq. (2.8) with
$a_1 = 1$



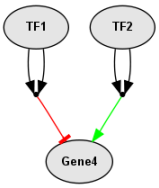

### Case 3: NAND logic



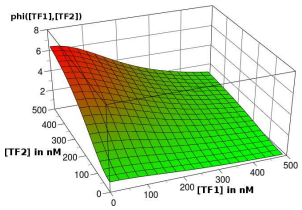

Eq. (2.5) with
$a_1 = 1$



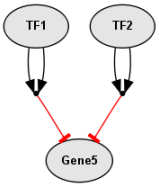

### Case 4: NOT TF1 AND TF2 logic




Eq. (2.5) with
$a_1 = -\infty$
$a_2 = 3$




### Case 5: NOT TF1 AND NOT TF2 logic




Eq. (2.5) with
$a_1 = -\infty$
$a_2 = -\infty$

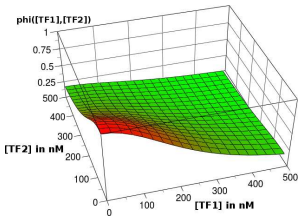



Figure 2.4 | **Response characteristic for various transcription models (cont.)**

computer. In a single window the parameters for the network and data generation can be specified. SynTReN generates a topology by extracting subnetworks from known biological networks (*Escherichia coli* or *Saccharomyces cerevisiae*). These networks show similar topological properties compared to the source networks, such as average in-degree and average path length. The gene regulation is only modeled on the transcription level, i.e., there is no protein level, by Michaelis-Menten and Hill kinetics with automatically chosen parameters. Other kinetics cannot be selected and the simulation parameters cannot be adjusted. The model system is solved to calculate normalized steady-state levels. The simulation time is linearly dependent on the number of genes in the network.

RMBNToolbox (Random Models for Biochemical Networks) was developed by Aho et al. (2007). The focus of this Matlab package is the (random) generation and simulation of biochemical systems. The toolbox provides functionalities to check the generated model on consistency and stability. Kinetics are randomly chosen from a kinetic law library. Nevertheless, the toolbox can also be used for modeling gene regulation on a simplified level. Transcription factor concentration is assumed to be equal to the concentration of the produced transcript and the transcription kinetics are based on the rate laws suggested by Mendes et al. (2003). Finally, time course data are then calculated and analysed within Matlab.

RENCO (REgulatory Network generator with COmbinatorial control) was developed by Roy et al. (2008). It is a command line tool for the generation of networks (scale-free, exponential degree, or user-defined) and the set-up of gene regulatory models, including an mRNA and protein layer, based on kinetics for the combinatorial control of transcription (Schilstra and Bolouri, 2002). The tool lacks an own simulation and visualization interface, but the model can be exported as SBML[6] and imported into other tools, where the simulation and visualization can be performed.

SynBioSS (Synthetic Biology Software Suite) developed by Hill et al. (2008) is a recently developed tool for modeling and stochastic simulation of synthetic genetic constructs. This tool guides the user in the design of multiscale biomolecular interaction models in high details. Models can be constructed with the help of the SynBioSS Designer. It is a Web-based component for the automatic generation of sets of biomolecular reactions given a construct composed of BioBrick standard biological parts[7]. The model can be imported to the SynBioSS Desktop simulator, which has to be installed on a local computer.

[6] SBML is short for Systems Biology Markup Language and is XML-based format for representing biological reaction networks (Hucka et al., 2003).

[7] BioBrick standard biological parts are genetically encoded functions that can be assembled into novel biological systems with defined properties (`http://www.partregistry.org`).

It is also possible to modify and specify biomolecular reactions by the user successively. The tool provides a large repository of reaction kinetics and parameters, stored in a Wiki database. Once a model system is defined, stochastic simulations are performed locally. Furthermore, it supports parallel execution of multiple trajectories, which are required for stochastic simulation to obtain a population distribution.

Another modeling and simulation tool is PyBioS developed by Wierling et al. (2007). The focus of this Web tool is the highly detailed description and simulation of small and large biochemical reaction systems. It is not restricted to gene regulation. A model can be constructed from various databases, such as KEGG (Kanehisa et al., 2008), Reactome (Matthews et al., 2009), or ConsensusPathDB (Kamburov et al., 2009), or by hand. Kinetics are assigned to each reaction automatically or have to be chosen from a kinetic repository. Furthermore, the kinetic parameters can be changed individually. An ordinary differential equation (ODE) system is generated automatically and solved numerically. Simulation results are visualized either as time course plots or within the network graph, which facilitates the interpretation of the results.

There are several other advanced software application similar to PyBioS, e.g., Gepasi (Mendes, 1993; 1997, Mendes and Kell, 1998), COPASI (Hoops et al., 2006), E-Cell (Tomita et al., 1999, Takahashi et al., 2003), Virtual Cell (Schaff et al., 1997, Loew and Schaff, 2001, Slepchenko et al., 2003), and Cell Designer (Funahashi et al., 2003) for the description, computation, and analysis of biochemical reaction systems. Most of these application define a model step by step and require manual work for entering all the model details. This can become cumbersome and error-prone in particular for large systems with many individual components.

The tools for forward modeling described above are either on a very simplified level, i.e., a high abstraction level, such as RMBNToolbox, or on a high detailed level, such as PyBioS, which are not restricted to gene regulation but needs much effort to define the model system as stated above. In the course of my thesis I have developed the Web application GEne Network GEnerator (GeNGe) to combine and extent features of the tools described above in a single framework.

GeNGe provides a user-friendly interface with rich functionalities for the computational characterization of gene regulatory networks. Network models with an appropriate description level of dynamical features are generated automatically and time course data under different perturbation conditions are calculated. In spite of the medium abstraction level, the modeling of different transcriptional dynamics are possible with GeNGe. Large sets of benchmark data for performance assessments can be easily collected within GeNGe.

The World Wide Web was selected for providing this software application, since it offers a flexible platform where no additional software has to be installed locally except a functional browser. Users benefit instantaneously from changes and new features in the application. Furthermore, the user does not need to pay attention to provide necessary computational power. All computations are performed on a computer cluster managed by the program server.

In the following section, I describe the application GeNGe in more details, including the user-interface and the different steps in the simulation process.

## 2.2 GEne Network GEnerator

In the course of my thesis I have designed and developed the modeling and simulation tool GEne Network GEnerator (GeNGe) for the systematic generation of gene regulatory networks and simulation of artificial expression data (recently published by Hache et al. (2009b)). GeNGe is implemented as an application for the Web application server Zope[8]. Therefore, the tool does not need to be installed on a local computer and is freely accessible for users over the Internet. GeNGe provides an interactive, user-friendly interface. The intuitive handling makes the modeling and simulation process easy. The user will be guided through each simulation step and can obtain additional information in the help system. Some technical aspects of GeNGe are given in Appendix C.

Within GeNGe, network topologies can easily be generated. These structures build the foundation for the modeling of gene regulatory networks. The corresponding mathematical models are set up with a specified kinetic schema, selected from a set of supported kinetic laws. Various *in silico* experiments can be performed, e.g., local perturbations, i.e., knock-downs of a certain degree, of a single gene or multiple genes. This can be done for an arbitrary number of networks in parallel to generate as many datasets as needed.

Moreover, GeNGe offers features for the topological characterization of gene regulatory networks. Network parameters are computed, such as in- and out-degree distributions, average path lengths, and clustering coefficients. Furthermore, by varying parameters of the kinetic laws or by choosing different kinetics, *in silico* analyses can be performed, e.g., on the effects of knock-downs (partial knock-downs) of a single gene or a group of genes. The results can be used to test network robustness, define critical network nodes and suitable candidates for perturbation experiments, and thus guide future experimental work.

[8] http://www.zope.org.

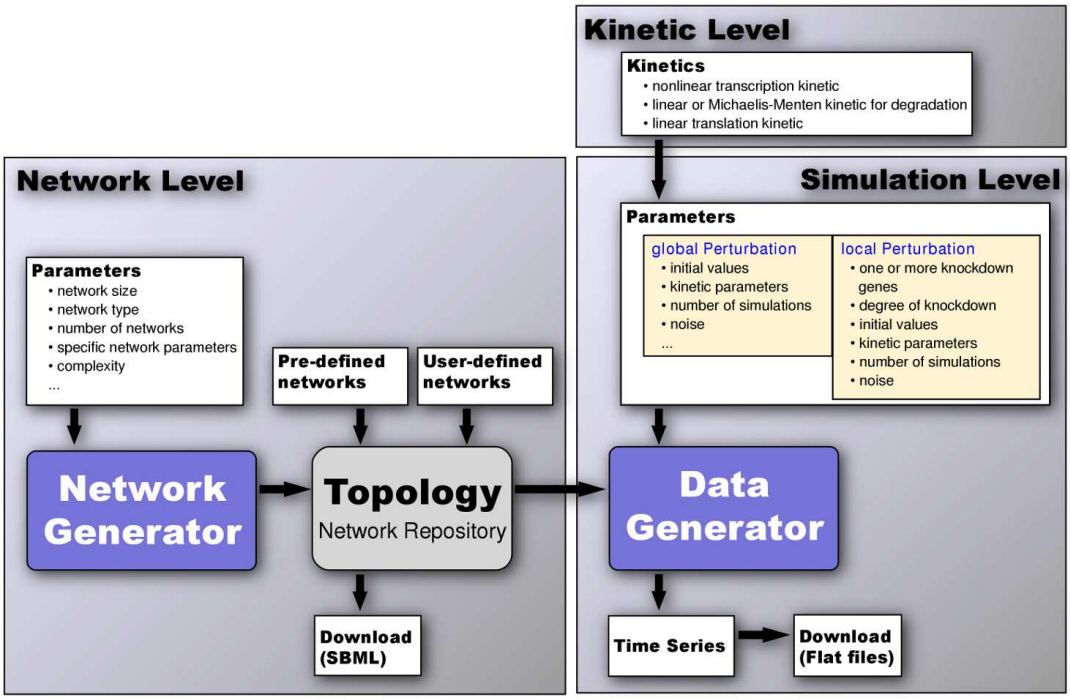



Figure 2.5 | **Flowchart of the simulation process in GeNGe.** It is divided into three levels, the network level, to generate a network topology, the kinetic level, to select kinetic laws of the dynamic model, and the simulation level, to set the parameter values and simulate time series with local or global perturbation.

The workflow of GeNGe is divided into three levels; the network level, the kinetic level, and the simulation level (Figure 2.5). In the first level, networks are added to a network repository. In the second level, the kinetic laws for the description of transcription, translation, and degradation are set. In the third level, the parameters and initial values are specified and the simulations are performed. In the following, each level is described in more details.

### 2.2.1 Network Level

Within the network level networks are generated that built the basis for modeling gene regulatory networks (Figure 2.6). Various network types are supported by GeNGe. An upload of user-defined networks given as adjacency matrices or tables of regulatory interactions is possible. Furthermore, a set of networks are pre-defined, e.g., artificial networks showing different dynamic characteristics, network motifs, or biological gene regulatory networks. Networks can be constructed or manipulated by adding, deleting, or changing edges in the network. In addition, different artificial types of networks, such

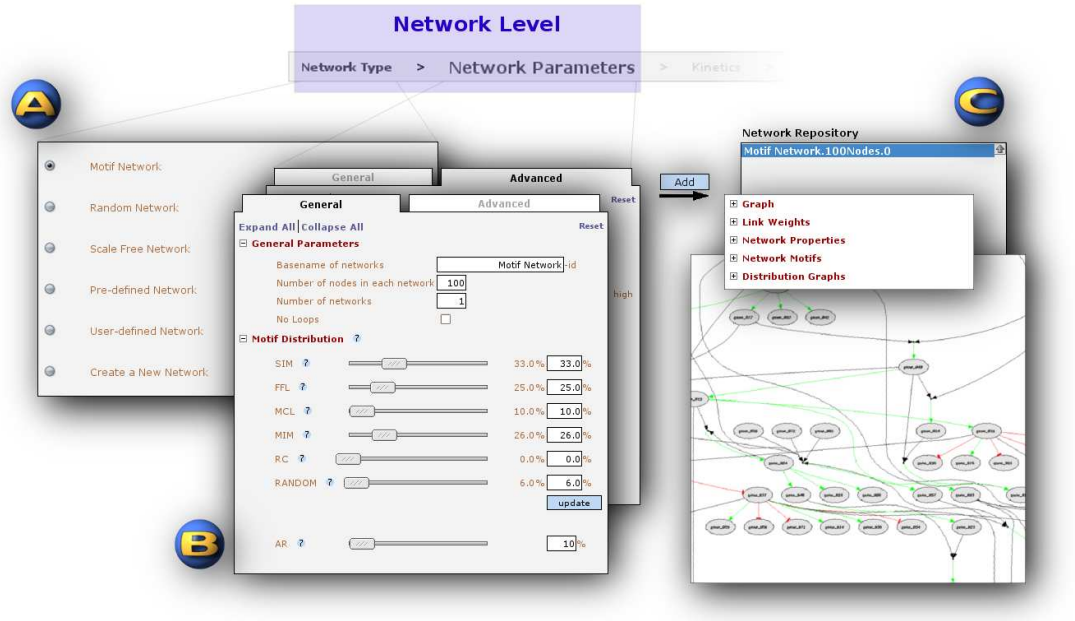



Figure 2.6 | **Network level of GeNGe. a** | Various network types are supported by GeNGe as a basis for modeling gene regulatory networks. **b** | After selecting a network type, corresponding network properties, e.g., network size, number of networks to be generated, or motif distribution for motif-networks, can be adjusted to obtain different network topologies. **c** | By pressing the *Add*-button, the network structures are added to the network repository, where the structure and various topological characteristics of each network can be inspected.

as motif networks, scale-free networks, and random networks can be generated automatically in GeNGe.

### Motif Networks

I have developed a new strategy to build networks which reflect biological features without explicit setting certain global properties, such as degree distributions. Biological networks show different properties compared to random networks, such as in- and out-degree distributions. Furthermore, several small network motifs, i.e., subnetworks showing certain characteristics, occur more often in gene regulatory networks than expected by chance. Different types of such motifs have been identified experimentally by Lee et al. (2002): single-input motif (SIM), feed-forward loop (FFL), multi-component loop (MCL), multi-input motif (MIM), regular chain (RC), and auto-regulation (AR) (see Appendix A). These motifs are used in GeNGe as basic building blocks in the motif network construction. During network generation motifs are assembled in the current network according to an adjustable motif distribution.

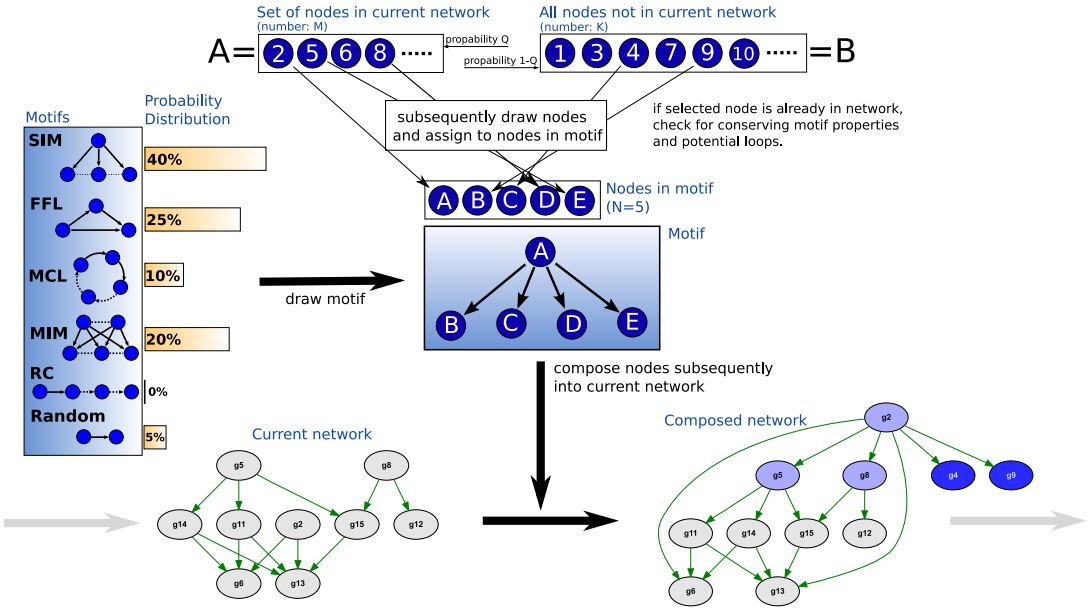



Figure 2.7 | **Motif network generation.** The motif network generation is an iterative process. Motifs are randomly drawn according to the user-defined motif distribution and composed into the network until the network has the desired size.

The motif network generation in GeNGe is an iterative process starting with an empty network (Figure 2.7). A motif is randomly drawn according to a user-defined motif distribution. Motif-specific properties, such as the number of regulators and targets, are set randomly and define a specific subnetwork, which is composed into the current network. Let $N$ be the number of nodes in the selected motif. This motif is composed into the current network by the following procedure. $N$ nodes are subsequently and randomly picked, either from a set of nodes in the current network (set $\mathcal{A}$) or nodes which are not already in the network (set $\mathcal{B}$). The complete set of all nodes in the current network is reduced by the nodes which cannot fulfill the functional role in the motif considering the node type in the motif (first regulator, second regulator, target, etc.) and the environments of the nodes in the current network. For instance, a node which is already a target (see, e.g., node *g6* in the example in Figure 2.7) cannot be a target in a SIM, since after adding a regulation the node would have more than one regulator, i.e., it does not have a single input, which is necessary in a SIM. Hence, the set $\mathcal{A}$ can be different for each node of the motif. In Appendix A on page 111 more information is given about each motif, the properties, and the constraints which are considered during the network construction process.

The random selection of a specific node either from $\mathcal{A}$ or $\mathcal{B}$ is a two stage random process. First, one of the two sets is picked randomly, where set $\mathcal{A}$

with size $M$ is selected with probability $Q$ and set $\mathcal{B}$ with size $K$ is selected with probability $1 - Q$. Afterwards, a specific node is drawn from the selected set with equal probability. That means, nodes from $\mathcal{A}$ have probability $Q/M$ and nodes from $\mathcal{B}$ have probability $(1 - Q)/K$ to be selected. The default value for $Q$ is $M/(M + K)$ resulting in a equal probability for all nodes, i.e., $Q/M = (1 - Q)/K$. This can be adjusted with a parameter $q \in [0, 1]$. In general, $Q$ is given by

$$Q(M, K|q) = \frac{q \cdot M}{q \cdot M + (1 - q) \cdot K}. \tag{2.9}$$

$q = 0.5$ yields to $Q = M/(M + K)$. The node selection with $q > 0.5$ favors nodes from $\mathcal{A}$, which results finally in a more interwoven network since nodes being already in the current network have a higher chance to be selected for motif composition. $q < 0.5$ has the opposite effect.

A selected node from $\mathcal{A}$ or $\mathcal{B}$ for a motif is immediately composed into the network, assuming it fulfills the above mentioned constraints. Thus, it belongs (after insertion) to the current network, which can be crucial for further selection of nodes for this network motif. The process of composing network motifs is repeated until all nodes are included in the network.

Generated motif networks show scale-free properties in the degrees and in the clustering coefficients. This indicates, that motif networks are scale-free and hierarchical. Network properties are discussed in Section 2.3 on page 54. For that, in Figure 2.15 several distribution plots are shown together with those of other networks and in Table 2.3 corresponding topological measures are listed.

#### Scale-free Networks

It is assumed that many real world networks are scale-free networks, i.e., they have a power law degree distribution in in- and out-degree. The probability that a node has $k$ in-links follows $P(k) \propto k^{-\gamma}$, where $\gamma$ is the degree exponent. This is also valid for the out-links with a different degree exponent (compare Section 1.3.2). This is indicated with a straight line in a log-log-distribution-plot of in- and out-degree. Bollobás et al. (2003) proposed a procedure to generate directed scale-free graphs, which I implemented in GeNGe. Similar to undirected scale-free graphs, a network grows under the assumption of preferential attachment, depending on in- and out-degree. That means, that a node which is added to the network prefers to link to more connected nodes represented by the degree of the nodes. The resulting network shows a power-law distribution of in- and out-degree. Several parameters have an influence on the resulting network structure and can be adjusted in GeNGe.

### Random Networks

Another network type which is implemented in GeNGe is the random network. Erdős and Rényi (1959) studied the properties of such networks where nodes are connected randomly with each other. An efficient algorithm by Batagelj and Brandes (2005) for the generation of large random networks is implemented in GeNGe. A random network growth starts with an unconnected graph of $N$ nodes and adds edges between them by chance. Each possible edge has the same probability and is independent of the other edges. This probability can be chosen arbitrarily in the interval [0,1]. In the final network the node degrees follow a Poisson-distribution. Hence, most nodes have approximately the same number of links.

### Pre-defined Networks

Various pre-defined networks are provided in GeNGe. Some of these networks show the possibility to model different transcriptional dynamics, such as saturation, switches, and oscillations. The corresponding kinetics are described in the kinetic level-section below. Furthermore, network motifs described by Lee et al. (2002) are implemented as small models. These models can serve as references for the investigation of dynamical properties of such motifs on a small scale.

Additionally to small artificial networks, a medium-scale biological network is implemented, which is a part of the developmental network in sea urchin described by Davidson et al. (2002). This model is based on many large-scale and individual perturbation analyses. It features some of the above mentioned motifs not isolated but with interconnections to each other. The complexity of the networks is characteristic for gene networks. Components which are active in endoderm during the 18-30 hours along with their interactions (activation or inhibition) are implemented in the model. The model contains 25 genes, most of them encode transcription factors with altogether 53 regulatory interactions (approximately 8% of all possible connections) as well as proteins and complexes. This model serves as medium-scale example of a realistic gene regulatory network and is analysed with several *in silico* perturbation experiments in Section 2.2.5. Furthermore, in the following Chapter the network is used for reverse engineering studies.

### User-defined Networks and Network Creation

GeNGe provides two additional possibilities to define a network. On the one hand, network structures can be uploaded as adjacency matrices or tables of

regulatory interactions, i.e., pairs of regulators and targets (Figure 2.8). A value has to be specified for each regulation where the sign defines the type of regulation, i.e., activation (positive value) or inhibition (negative value), and the absolute value gives the regulation strength that is used as a kinetic parameter. Furthermore, joint interaction sets can be specified in a table format. On the other hand, networks can be constructed by the user from scratch. Nodes and edges can be added subsequently to a network and the regulation strengths can be specified.

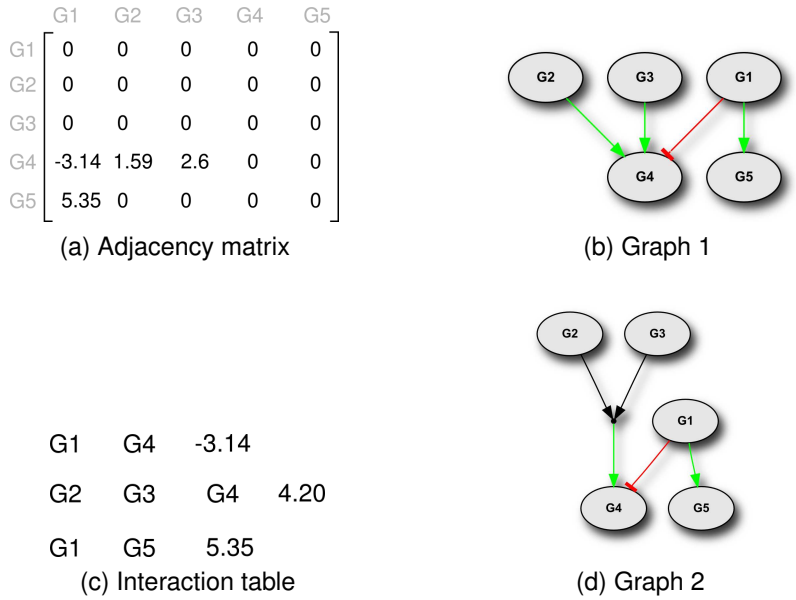



Figure 2.8 | **User-defined Networks.** **a** | Adjacency matrix is a topology representation with additional information about regulation strengths (for the kinetics). **b** | Corresponding network graph. Only the sign of regulation strengths are considered, where positive and negative values are represented as green edges (activation) and red edges (inhibition), respectively. **c** | If there are joint interactions, then the network is represented with an interaction table, where each line corresponds to an interaction. The last given node in a line is regulated by the other nodes with the given regulation strength. **d** | Corresponding graph with joint interaction sets. Nodes in a joint interaction set are linked to a joint node with black edges.

Moreover, all created networks in the network repository can be changed, copied and removed in every simulation level. Networks with slightly different structures can be constructed and analysed in parallel.

### Network Properties and Motif Search

GeNGe offers features for the topological characterization of gene regulatory networks. First, the topological structure of each network can be visually inspected. A node in a graph represents simultaneously a gene and the corre-

sponding mRNA and protein. A directed link from one node to another indicates a direct regulatory effect of the transcription factor coded by the gene of the first node on the transcription of the gene represented by the second node. The link color depicts the type of the effect, i.e., green for activation and red for inhibition. Transcription factors acting jointly, i.e., members of joint interaction sets, which were introduced above, are linked to a joint point by a black edge. This point is further connected to a target gene by a colored directed link according to the interaction type of the joint interaction set.

Moreover, various topological measures are shown and computed, such as number of nodes, interactions, regulators, targets, input- and output nodes[9]. Furthermore, averaged in-degree, out-degree, path-length, and clustering-coefficient are calculated. With these parameters and the provided distribution plots of in- and out-degree, average path-lengths, and clustering-coefficients the network topology can be investigated. In Section 2.3 various topological measures are calculated of several networks which are generated in GeNGe. Their properties are compared to a gene regulatory network for human extracted from TRANSFAC.

Furthermore, motif search is implemented for a different view on the topology. It is shown, that biological networks contain statistically significant patterns serving as indicators of biological functions (Lee et al., 2002, Milo et al., 2002, Shen-Orr et al., 2002). The implemented search algorithm seeks in the complete network for such subnetworks with pre-defined properties. These properties of the individual motifs are described in Appendix A in more detail. The algorithm does not calculate a statistical significance for the occurrence of a motif compared to the occurrence in random networks like Berg and Lässig (2004). It rather counts how often certain subgraph templates, which are already identified as significant in biological networks, occur in the network. The size of a particular template of a certain motif is not fixed. For instance, the number of targets in a single input motif is two or larger. The algorithm tries to find the largest subnetwork consistent with a given motif pattern. Different motifs can overlap, but a node can have different functions in different motifs.

### 2.2.2 Kinetic Level

Given a specific network structure, GeNGe provides a mathematical model for the regulatory interactions. This model covers the transcriptional as well as the translational layer. These layers include instances for mRNAs and proteins as well as for polymerase and ribosome. This is discussed in Section 2.1.1 on page 24 as a simplified model of gene regulation.

[9]Input nodes do not have any in-links and output nodes do not have any out-links in the network.

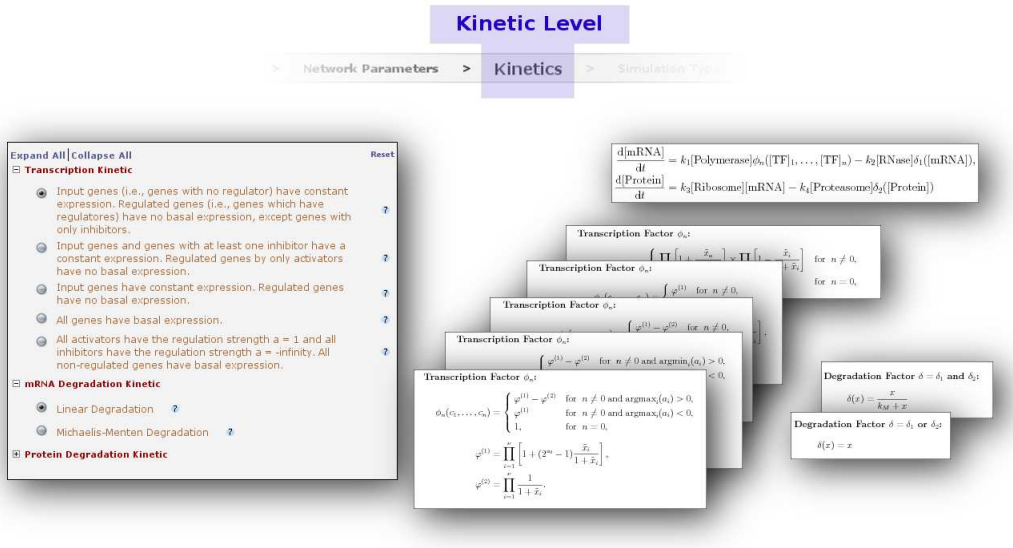



Figure 2.9 | **Kinetic level of GeNGe.** Various kinetics for the description of transcription and degradation are supported. The translation is described by a linear kinetic which cannot be changed. In contrast, several non-linear transcription kinetics are provided. Moreover, a linear and a Michaelis-Menten-like kinetic can be selected for degradation, independently for the mRNAs and proteins. The selected kinetic schema is applied for all mRNAs and proteins in the model. Nevertheless, kinetic parameters can be adjusted individually in the next level.

The dynamic of the system is described by the rate equations Eq. (2.1a) and Eq. (2.1b). In Eq. (2.1b) the translation is described by the first term on the right hand side with a linear kinetic. Usually, degradation is modeled as well with a linear kinetic. However, it is observed, that protein degradation can also be described by a Michaelis-Menten kinetic (Grilly et al., 2007). Therefore, GeNGe supports a linear- ($\delta(x) = x$) and a Michaelis-Menten-like ($\delta(x) = x/(K_M + x)$) degradation for mRNAs and proteins, where $x$ is the respective mRNA or protein concentration. An additional parameter $K_M$, specific for each mRNA and protein is introduced for the Michaelis-Menten kinetic, representing the substrate concentration, i.e., mRNA or protein concentration, at which the reaction rate reaches half of its maximum value. The degradation kinetic schema for mRNAs and proteins can be chosen independently.

Internally, a rescaled equation system is used to avoid very large values and to weight concentrations differently in the kinetics. For that, Eq. (2.1a) and Eq. (2.1b) are partially non-dimensionalized by rescaling the concentrations

$$[\mathrm{mRNA}] = M \cdot [\mathrm{mRNA}]' \qquad [\mathrm{Protein}] = P \cdot [\mathrm{Protein}]', \tag{2.10}$$

where $M$ and $P$ are normalization factors. The normalized concentrations $[\cdot]'$ are then dimensionless. Following the approach by Elowitz and Leibler (2000), the normalization factor $M$ of the mRNAs can be interpreted as the translation

efficiency that is the average concentration of proteins produced per concentration unit of mRNA. The protein concentration normalization factor can be considered as the concentration to half-maximally activate or repress the gene transcription, while other transcription factor concentrations are considered as constant. Under the assumption of independently binding transcription factors the normalization constants are equal to the normalization constants, **K**, introduced above. If a protein does not act as a transcription factor, the normalization is an artificial scaling factor and has only an impact on the scale of the time course values for this particular protein.

The kinetic reads with the scaling factors

$$\frac{\mathrm{d}[\mathrm{mRNA}]'}{\mathrm{d}t} = k_1'[\mathrm{Polymerase}]\phi_n(P\cdot c_{t_1}', \ldots, P\cdot c_{t_n}') - k_2'[\mathrm{RNase}]\delta(M\cdot[\mathrm{mRNA}]'), \tag{2.11}$$

$$\frac{\mathrm{d}[\mathrm{Protein}]'}{\mathrm{d}t} = k_3'[\mathrm{Ribosome}][\mathrm{mRNA}]' - k_4'[\mathrm{Proteasome}]\delta(P\cdot[\mathrm{Protein}]'), \tag{2.12}$$

with the rescaled kinetic constants

$$k_1' = \frac{k_1}{M} \qquad k_2' = \frac{k_2}{M}\ , \tag{2.13}$$

$$k_3' = \frac{M}{P}k_3 \qquad k_4' = \frac{k_4}{P}. \tag{2.14}$$

Each normalization factor of the mRNAs and proteins can be specified within GeNGe independently. Eq. (2.11) and Eq. (2.12) are numerically solved in the simulation level by GeNGe and the time series output is rescaled to the input dimension. For that, the enzyme concentration and all parameters are assumed to be constant over time.

Table 2.2 | **Units in GeNGe for the variables and parameters.**

| **Component** | **Unit** |
|---|---|
| functions $\phi_n$ and $\delta$ | 1 |
| unnormalized concentrations of mRNAs and proteins | 1 nM |
| normalization constants $M$ and $P$ | 1 nM |
| kinetic constants $k_1, k_2, k_4$ | $1\,\mathrm{min}^{-1}$ |
| kinetic constant $k_3$ | $1\,\mathrm{nM}^{-1}\,\mathrm{min}^{-1}$ |
| Michaelis constant $K_M$ | 1 nM |

In GeNGe all variables and constants are assigned a unit (Table 2.2). The concentration units can be converted into numbers of entities by means of a

volume. The volume of a cell is approximately $V \approx 2\,\text{fL}$ for a typical bacterium and $V \approx 4\,\text{pL}$ for a typical mammalian cell which correspond to a sphere diameter $d \approx 1.56\,\mu\text{m}$ and $d \approx 19.69\,\mu\text{L}$, respectively (Alberts et al., 2008). With the Avogadro constant, $N_A$, of approximately $6.0221 \times 10^{23}\,\text{mol}^{-1}$, which is the number of elementary entities in one mole, the concentration value $c$ can be converted to a number of entities by $V \cdot N_A \cdot c$. For instance, in a bacterium cell a concentration of $c = 1\,\text{nM}$ would corresponds to approximately 1 and in a mammalian cell to over 2400 entities.

### 2.2.3 Simulation Level

All kinetic parameters and initial values can be specified in the last level before the simulations are performed (see Figure 2.10). The unnormalized initial values for each component in the model can be chosen arbitrarily or set randomly. To simulate global perturbations (for example network noise) it is also possible to choose the initial values randomly from a Gaussian distribution centered at the corresponding steady-state concentrations. The steady state is determined numerically in a pre-simulation. The end time of such a simulation can be set to a large value to ensure, that changes of concentrations do not occur anymore.

Targets of local perturbations, i.e., knock-outs or knock-downs of certain degrees can easily be selected. Moreover, multiple simulation runs with sets of such perturbation targets can be prepared. For instance, it is possible to knock-out each regulator in a model in separated simulation runs.

Moreover, each kinetic parameter can be changed individually. Specific perturbations of a gene or set of genes, are realized by reducing the corresponding transcriptional rate, $k_1$, in Eq. (2.1a), according to the specified perturbation level (partial knock-downs). For a 100% knock-down, the transcription rate factor is set to zero, hence, the particular gene will not be transcribed but degraded. Therefore, the effect of RNA interference can be modeled. All other parameters and initial values for the control and the knock-down runs are the same.

Based on the network topology, the kinetics, and the kinetic parameters an ordinary differential equation (ODE) system of the network is set up and exported to an ODE solver by Web services API provided by PyBioS (Wierling et al., 2007). See Appendix for detailed information about the connection of GeNGe and PyBioS.

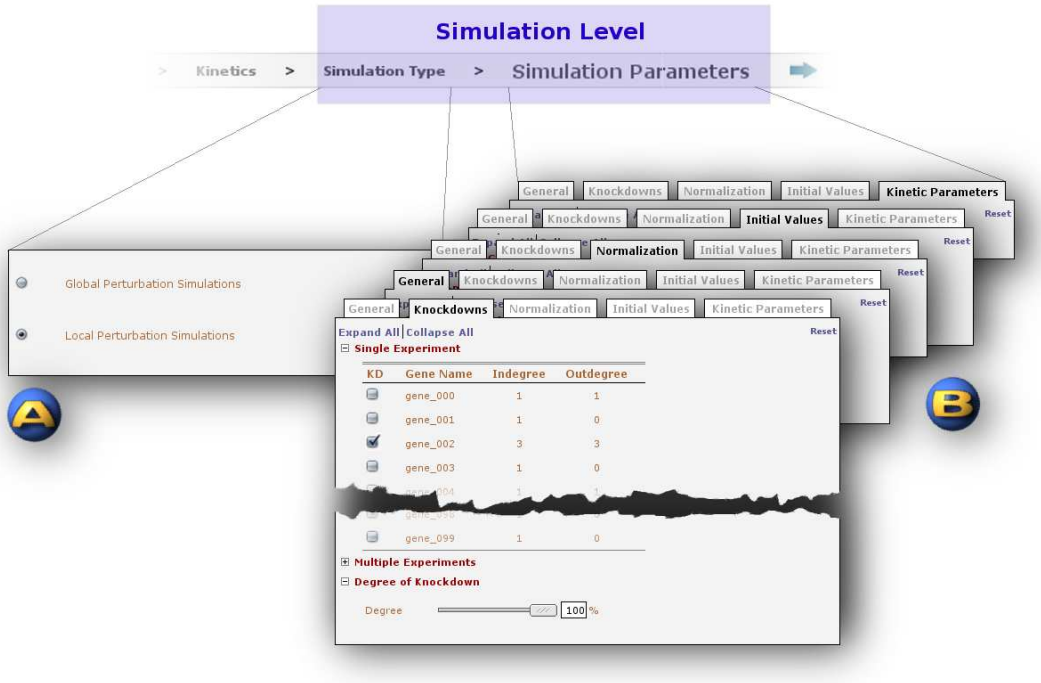



Figure 2.10 | **Simulation level of GeNGe. a** | Two types of simulations are supported, global and local perturbations. In the first simulation type, all genes are targets of a perturbation. The latter simulation type provides an additional interface for selecting certain genes for specific perturbation. **b** | Corresponding kinetic and simulation parameters can be specified, such as number of simulations, simulation time, initial values, normalization constants, rate constants, and knock-down genes.

### 2.2.4 Simulation Results

The highly non-linear differential equation systems which are set up for each previously defined simulation are solved numerically. The calculations are independent of each other, in particular, the values of parameters and initial values defined as random are drawn independently from the specified random distribution. In contrast, in local perturbation simulations a control and the corresponding treatment state differ only in the parameters defining the knock-down. All other parameters and initial values are identically set between control and treatment state.

All simulation results, including the network structure as a matrix, the mathematical model in SBML format, and the mRNA and protein concentration time series obtained in each *in silico* experiment can be downloaded or visually inspected within GeNGe (Figure 2.11). The time course data are shown in plots for each component. Furthermore, for local perturbation simulations the resulted $\log_2$ ratios of treatment vs. control of mRNA concentration values at the last time point are visualized by ratio dependent colors in the network

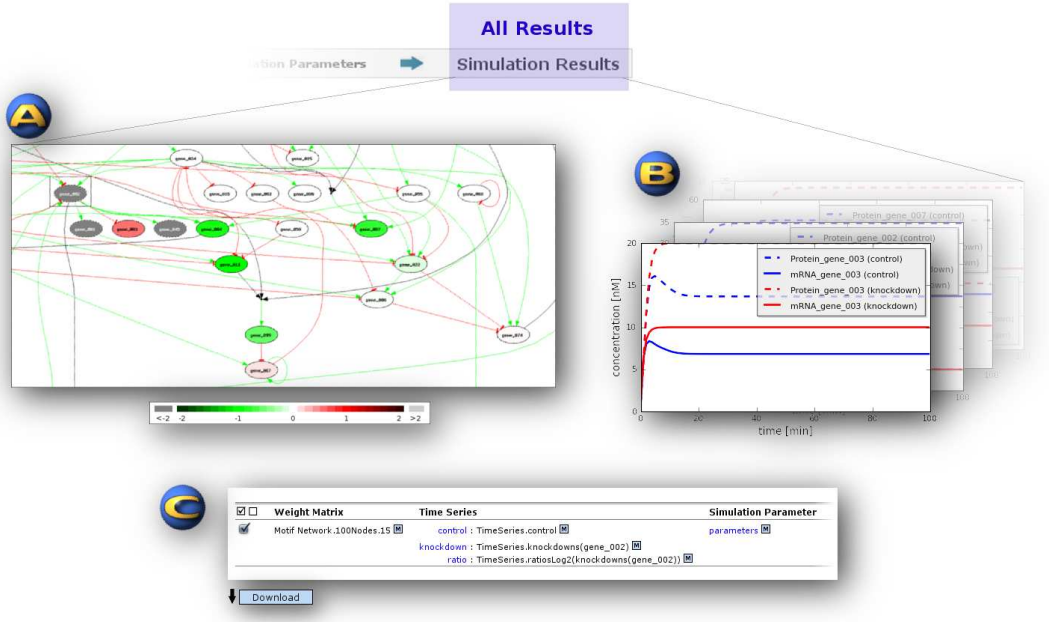



Figure 2.11 | **Simulation results in GeNGe. a** | For local perturbations, the resulted $\log_2$ ratios of treatment vs. control of the last mRNA concentration values are visualized by colors in the network graph. Red shaded nodes correspond to up-regulated, light grey to strongly up-regulated, green shaded to down-regulated, and dark grey strongly down-regulated genes. Knock-down genes are indicated by a rectangle around corresponding nodes. White nodes have the ratio 1, hence they are unaffected. **b** | Time course data are plotted for each component, where blue solid lines represent mRNA concentrations and dashed lines corresponding protein concentrations. Treatment curves are plotted in red, if available. **c** | All simulation results are collected and provided for download.

graph. Such a colored graph helps to identify rapidly affected genes by local perturbations and study the connections between them.

The procedure of simulations can be repeated with different graphs and parameter settings and used in an iterative way. Selected data are provided for download as compressed tar-archives. The parameter settings are stored as well and can be imported again into GeNGe to repeat or perform new simulations at a later date.

### 2.2.5 Forward Modeling examples

Specific types of transcription dynamics, such as saturation, bistability, and oscillation have been found in biological gene regulatory systems, e.g., a genetic switch in $\lambda$-bacteriophage (Ptashne, 2004), a galactose genetic switch in yeast (Platt and Reece, 1998), a circadian oscillator in cynaobacteria (Ishiura et al., 1998), and a segmentation clock in vertebrates (Goldbeter and Pourquié, 2008).

Such dynamics are reproducible with the developed model of gene regulation in GeNGe. Moreover, larger gene regulatory models can be used for predicting perturbation effects.

### Small Models of Gene Regulation

To demonstrate, that GeNGe is capable of modeling such transcriptional dynamics, several examples are given below. A two node example with additional nodes giving input signals behaves like a switch between two states, initiated by an input signal (Figure 2.12). The input stimulus forces the switch into one state or the other and remains in that state even the stimulus has been removed. Such a bistable switch is well studied, e.g., from a theoretical perspective by Cherry and Adler (2000) or *in silico* and *in vivo* as a genetic construct in *Escherichia coli* by Gardner et al. (2000).

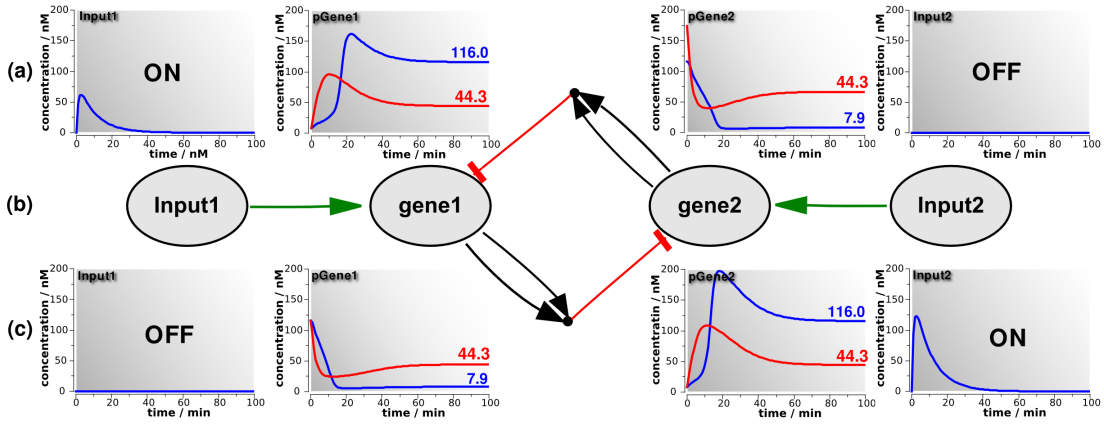



Figure 2.12 | **Toggle switch model**. A two gene model with two input signals. Blue and red curves indicate protein concentrations for cooperatively and non-cooperatively binding repressors, respectively. The steady state values are depicted in the plots. Two signal states are considered. **a** | Input1 is turned on for a short while, Input2 is off. **b** | The network graph. **c** | Input1 is off and Input2 is turned on.

The switch model of two identical genes and two input signals shown in Figure 2.12 is described by

$$\phi_i(x_j) = \left[1 + \frac{[\text{Input}_i]}{1 + [\text{Input}_i]}\right]\left[1 + (2^{a_{ij}} - 1)\frac{x_j^n}{1 + x_j^n}\right] \qquad (i,j) \in \{(1,2),(2,1)\} \quad (2.15)$$

where $(x_1, x_2)$ are the protein concentrations corresponding to the genes, $Input_1$ and $Input_2$ are the input signals, and $n$ is a cooperativity exponent. This model

exemplifies that to obtain bistability the inhibitors must be repressors with cooperative binding characteristics, i.e., $n > 1$. Two situations are studied, *Input1* ON and *Input2* OFF on the one hand, shown in the first line of plots and *Input1* OFF and *Input2* ON shown in the second line. The red and blue curves represent the protein concentrations corresponding to the genes considering non-cooperative binding ($n = 1$) and cooperative binding ($n = 2$), respectively. In the case of non-cooperative repression, both protein concentrations relax to the same value after the perturbation of one of the inputs. In contrast, in the case of cooperative binding (blue lines) the first input signal pushes *gene1* to a different state than *gene2* (116.0 nM compared to 7.9 nM respective protein concentrations, i.e., a fold change of over 14) and the second input signal can switch the state for which *gene2* is turned on and *gene1* is turned off.

Another example is an oscillator based on a remarkably simple three-gene model designed *in silico* and constructed in *E. coli* by Elowitz and Leibler (2000). This model shows oscillatory behavior of the protein concentrations. They termed it *Repressilator* since it consists of three repressible promoters arranged in a cycle. The authors claimed, that the kinetic parameters, such as protein synthesis and degradation rate have to be selected carefully to obtain oscillations in the concentrations. They found that oscillations are favored by high transcription and translation rate, comparable protein and mRNA degradation rates, cooperative repression characteristics, and efficient repression. Considering these theoretical results, Elowitz and Leibler selected appropriate strong and tightly repressible hybrid promoters for the construction of the model in *E. coli*. Furthermore, a reduction of the protein-lifetime was realized by using small stable RNA (ssrA) tags. With this, they could establish temporal oscillations in the protein concentrations *in vivo*.

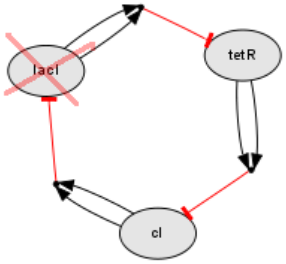
The implemented three-gene model in GeNGe with six molecular species

$$\phi^{\mathrm{lacI}}([\mathrm{Protein_{cI}}]) = 1 + (2^{a_{\mathrm{cI}}} - 1)\,\frac{[\mathrm{Protein_{cI}}]^2}{1 + [\mathrm{Protein_{cI}}]^2} \tag{2.16a}$$

$$\phi^{\mathrm{tetR}}([\mathrm{Protein_{lacI}}]) = 1 + (2^{a_{\mathrm{lacI}}} - 1)\,\frac{[\mathrm{Protein_{lacI}}]^2}{1 + [\mathrm{Protein_{lacI}}]^2} \tag{2.16b}$$

$$\phi^{\mathrm{cI}}([\mathrm{Protein_{tetR}}]) = 1 + (2^{a_{\mathrm{tetR}}} - 1)\,\frac{[\mathrm{Protein_{tetR}}]^2}{1 + [\mathrm{Protein_{tetR}}]^2} \tag{2.16c}$$

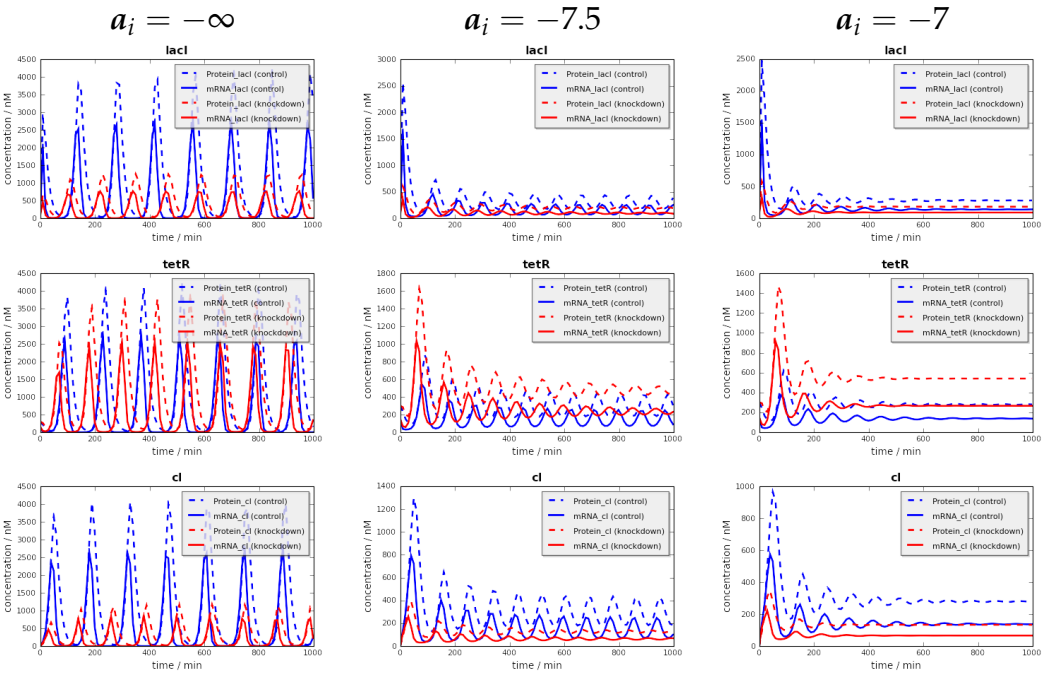
is inspired by the repressilator proposed by Elowitz and Leibler (2000) but uses slightly different transcription functions, Eq. (2.16). The model is used to investigate the impact of differently strong inhibitors and a knock-down on the oscillatory behavior. The strength of an inhibitor is represented by the parameter $a_i < 0$, where a decreasing $a_i$ results in an increasing inhibitory effect. The model shown in Figure 2.13 with the given parameters leads to oscillations in




(a) Network Graph.

| Parameter | Value |
|---|---|
| $k_1$ | $1730.1\,\mathrm{min}^{-1}$ |
| $k_2$ | $0.35\,\mathrm{min}^{-1}$ |
| $k_3$ | $0.14\,\mathrm{nM}^{-1}\,\mathrm{min}^{-1}$ |
| $k_4$ | $0.069\,\mathrm{min}^{-1}$ |
| $M$ | $20\,\mathrm{nM}$ |
| $P$ | $40\,\mathrm{nM}$ |

(b) Kinetic Parameters




(c) Time course plots of the oscillator.

Figure 2.13 | **Oscillator.** The oscillatory model was adapted from the Repressilator by Elowitz and Leibler (2000). **a** | Network Graph of the oscillatory model. The gene *lacI* is a target of an 80% knock-down. **b** | Kinetic parameters and normalization constants of the model system Eq. (2.1a), Eq. (2.1b), and Eq. (2.16) are identical for each component. **c** | Time course plots (blue) of mRNA concentration (solid lines) and protein concentrations (dashed lines) of each component are shown. Red curves represent the time curves for a 80% knock-down of *lacI* and blue curves represent the control. Time course data were simulated for different regulation strengths $a_i$ (columns) for $i = \mathrm{lacI}, \mathrm{tetR}, \mathrm{cI}$ (rows) for the system given by Eq. (2.16). A stable oscillation is observable for the strongest inhibitor $a_i = -\infty\ \forall i$. In contrast, the oscillation breaks down for a relatively weak inhibitor with $a_i = -7\ \forall i$. In a simplified range ($a_i = -7.5\ \forall i$), a stable oscillation can be observed which breaks down after knock-down.

the concentrations of mRNAs and proteins. The kinetic parameters and the normalization constants are identical for corresponding molecular species.

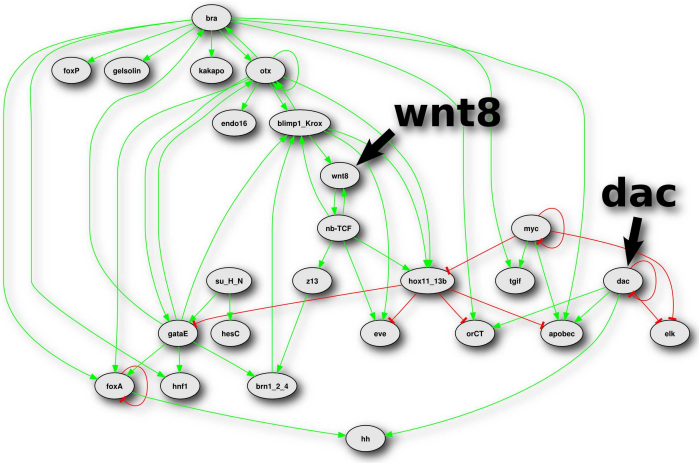
The system shows very robust oscillations for the strongest inhibitors with $a_i = -\infty$ for $i = \text{lacI}, \text{tetR}, \text{cI}$. The protein concentration (dashed lines) follow the mRNA concentrations with a small delay. The system continuous oscillating even after an 80% knock-down of gene *lacI*, but with a smaller amplitude and a higher frequency. Also, decreasing the strength of the inhibitors ($a_i = -7.5$; second column in Figure 2.13) results in an increase of the oscillation frequency with a smaller amplitude. A weaker inhibitor is not able to repress the target gene to full extent, which in turn means, that this gene has a relatively stronger expression. Moreover, the corresponding protein of this gene can repress its target gene sooner. Hence, the expression level of this gene does not reach such a high level as in the case of stronger inhibitors and decreases earlier. Therefore, the frequency of the oscillations is higher with a smaller amplitude. However, a total breakdown of the oscillations is observable after an 80% knock-down of gene *lacI* (later than 1 000 minutes, not shown in second column in Figure 2.13).

The inhibitory system with $a_i = -7 \; \forall i$ (third column in Figure 2.13) shows no continuous oscillations and relaxes to a stable steady state after some time. The inhibition effect of the transcription factors are not strong enough to repress their respective target genes to give the next genes in the circle the possibility to be expressed at a sufficient level. The amplitudes are getting smaller until the steady state is reached.

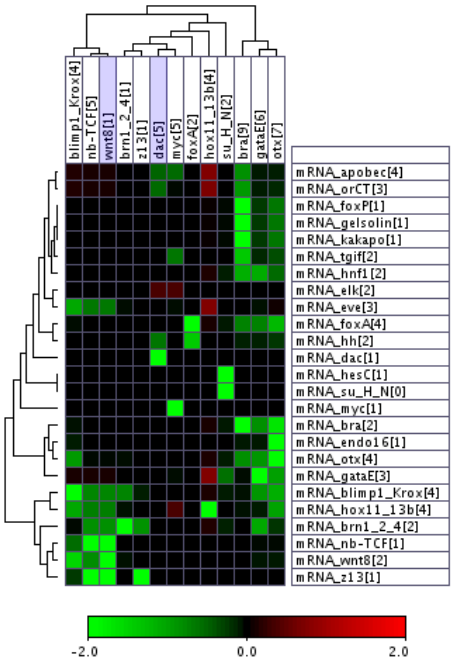
The influence of differently strong inhibitors on the oscillations could be verified. A further conclusion of this simulation study of the oscillatory system is that reducing the transcription rate results in an increase of the oscillation frequency with a reduced amplitude. Furthermore, an increase of the amplitude can be obtained by considering a stronger cooperative binding.

#### Perturbations in a Biological Network

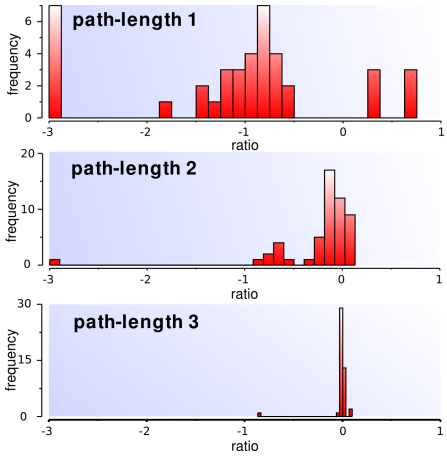
Whereas the effect of local perturbations is often straightforward in small networks, in larger networks the impact of such perturbations on the steady-state of the system is less obvious and thus simulations can guide experimental work in selecting the most promising candidates. The third gene regulatory model example is given by the developmental network shown in Figure 2.14 with the underlying kinetic given in Eq. (2.8). Each gene encoding a transcription factor in the model is knocked out (100% knock-down) and the $\log_2$-ratios compared to the control state are visualized in Figure 2.14b. It can be observed that knock-outs of genes encoding highly connected transcription factors with many targets do not have always a large impact on the global system state, such




(a) Graph




(b) Knock-outs




(c) Ratio distribution

Figure 2.14 | **All single knock-outs in developmental network. a** | A part of the developmental gene regulatory network described by Davidson et al. (2002) is shown (version from October, 3rd 2008). Green edges indicate regulatory activations, red edges inhibitions. **b** | Knock-out experiments of each gene encoding a transcription factor in developmental network. Visualization of $\log_2$-ratios of mRNA (knock-down vs. control states) for each transcription factor (column with out-degree) and each gene (row with in-degree). Green colors indicate down-regulation and red colors indicate up-regulation. **c** | The distribution of all ratios of genes with given path length to knocked-out gene are shown. All knock-out experiments are considered. The ratio of a complete knock-out is set to $-3$. The direct target genes of knock-outs are neglected.

as in the case of *dac*. In contrast, for example in the case of *wnt8*, transcription factors with critical positions within the network can have a large downstream effect even if they have only a few direct targets.

The knockdown effect of an arbitrary gene does not spread into the network very deep. The effect nearly vanishes after a path-length three (Figure 2.14c). The ratios of knock-down vs. control state are largest for direct targets of knocked-out genes and increases rapidly with increasing distance to the initial perturbation. Hence, single knock-outs have only local effects. Furthermore, a complete knock-out of the system cannot be achieved with a single knock-out, therefore, the network is robust against single perturbations. With multiple knock-outs of highly connected genes, the system will break down.

## 2.3 Network Properties

Examples of a random network, scale-free network, and motif-network are generated and compared to the human gene regulatory network extracted from TRANSFAC (version 11.3). The TRANSFAC database contains curated experimental data about regulatory elements in gene expression. To each reported transcription factor additional information is given, such as names, symbols, type, species where it is has been identified, encoding gene, homologs, sequences, literature references, and links to TRANSFAC tables and other databases. Most interesting with respect to gene regulatory networks are information about binding sites and regulated genes of transcription factors. Each such interaction is a quality on a six level scale assigned. For some transcription factors, there are binding sequences given but without a link to a target gene. These are often artificial or consensus sequences. Furthermore, there are factors found in one species with reported binding sites in other species due to homology. I extracted all transcription factor proteins (I neglected miRNAs and other non-protein factors) and their target genes from the TRANSFAC database, if the transcription factor or its target gene belongs to human.

Distribution plots associated to the random, scale-free, motif, and TRANSFAC network are shown in Figure 2.15. Additionally, several topological measures are calculated and listed in Table 2.3. All these networks have the same number of nodes and a similar number of edges. The expected distributions of in- and out-degree as well as clustering coefficient for a random and a scale-free network can be observed. In Section 1.3.2 it was mentioned that random networks have a Poisson-distribution in in-degrees and out-degrees. In the shown random network these features are not well distinctive, since a low probability, $p = 0.001$, for a connection between two arbitrary nodes was used for the network generation. This value of $p$ was selected to obtain similar interactions

Table 2.3 | **Topological measures of generated networks.** Measures are calculated for respective examples of a random network, a scale-free network, and a motif network. All these networks have similar number of nodes and edges as the human gene regulatory network from TRANSFAC. Respective distribution plots are shown in Figure 2.15. The measures are from top: $N$, number of nodes in the network; number of input nodes (nodes without in-links); number of output nodes (nodes without any out-link); number of clusters (connected subnetwork without any links to other clusters); number of interactions (edges); $\langle k \rangle$, average in- and out-degree; $\langle l \rangle$, averaged shortest path length; $\langle C \rangle$, average clustering coefficient; $\gamma_{\text{in,out,clust}}$, exponent of fitting curve $f(k) = a \cdot k^{-\gamma}$ (omitted for random networks); number of motifs of SIM (single-input motif), FFL (feed-forward loop), MCL (multi-component loop), MIM (multi-input motif), and AR (auto-regulation).

| Measure | Random Net. | Scale-free Net. | Motif Net. | TRANSFAC Net. |
|---|---|---|---|---|
| $N$ | 1644 | 1644 | 1644 | 1644 |
| # input nodes | 302 | 380 | 293 | 849 |
| # output nodes | 260 | 739 | 1072 | 770 |
| # clusters | 4 | 1 | 19 | 33 |
| # interactions | 2758 | 2631 | 2765 | 2960 |
| $\langle k \rangle$ | 2.055 | 1.600 | 2.047 | 1.800 |
| $\langle l \rangle$ | 12.545 | 7.360 | 5.15 | 1.219 |
| $\langle C \rangle$ | 0.001 | 0.003 | 0.031 | 0.002 |
| $\gamma_{\text{in}}$ | - | 2.194 | 2.576 | 1.859 |
| $\gamma_{\text{out}}$ | - | 1.630 | 1.497 | 1.437 |
| $\gamma_{\text{clust}}$ | - | 0.065 | 3.428 | 0.683 |
| SIM | 63 | 151 | 84 | 54 |
| FFL | 4 | 187 | 60 | 44 |
| MCL | 438 | 241 | 18 | 0 |
| MIM | 1 | 236 | 62 | 818 |
| AR | 4 | 4 | 0 | 10 |

compared to the other networks. However, shorter path lengths and a Poisson-distribution in the degrees can be observed for larger values of $p$.

The scale-free network shows the expected scale-free property in the degrees and a constant distribution in the clustering coefficient ($\gamma_{\text{clust}} \approx 0$). Furthermore, the shortest path length distribution is shifted to shorter values compared to the random network, i.e., a smaller average path length, $\langle l \rangle$.

Also the motif network has a scale-free property. Additionally a power-law distribution in the clustering coefficient is observable that is in indicator for hierarchical networks. This is remarkable, since no constraints on the degrees are specified in the network generation explicitly. Biological networks are assumed to have such a hierarchical property. The TRANSFAC network has similar properties of the motif network. However, TRANSFAC is a collection of a multitude of experimental observations of gene regulation. It consists of many separated subnetworks and is not very deep, i.e., there are only a few long paths in the network. Therefore, the shortest path-length distribution has a high peak at a very low value and the average shortest path length is $\langle l \rangle \approx 1.2$.

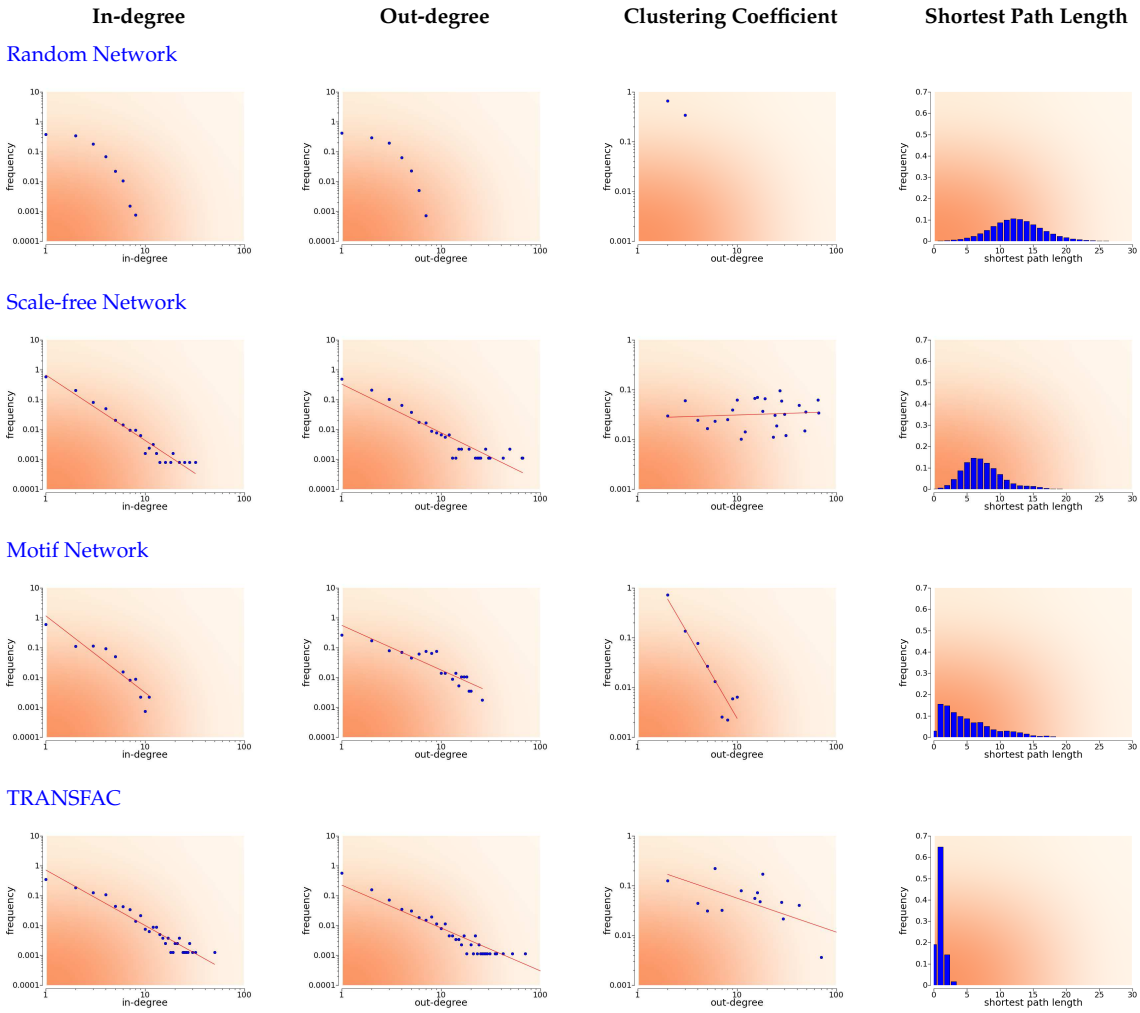



Figure 2.15 | **Generated networks.** All networks have the same number of nodes and a similar number of edges. Examples of a random network, scale-free network, and a motif-network are shown. Furthermore, the humans gene regulatory network from TRANSFAC is extracted. Various distribution plots are shown in a $\log - \log$-scale. Additionally, the shortest path-length distribution is shown.

# CHAPTER 3

# GNRevealer – a Neural Network Approach for Reverse Engineering

A complement to forward modeling is reverse engineering. The purpose of reverse engineering is to reconstruct the unknown gene regulatory network from given data. For that, gene regulatory network models based on neural networks can be used. Neural networks are well suitable for modeling gene regulation with only a few parameters. The dynamics are given with a set of ordinary differential equations. They provide the possibility for modeling dynamical processes with multigenic regulation including activation, inhibition, and self-regulation described by non-linear kinetics. These features are not always given in other gene regulation models of reverse engineering approaches. In this chapter I present a reverse engineering method called GNRevealer that is based on neural networks (Hache et al., 2007). I have adapted a learning strategy for gene regulatory networks for finding an optimal parameter set which explains the given temporal expression profiles best.

## 3.1 Model

For a system of $N$ components a set of first-order ordinary differential equations,

$$\frac{\mathrm{d}y_i(t)}{\mathrm{d}t} = f_i(\mathbf{y}(\mathbf{t})) \qquad \forall i \in \mathcal{N}, \tag{3.1}$$

may be considered to describe the temporal dynamics of an arbitrary systems. This general concept can also be applied in order to describe gene regulation. In such models, to each node, $i$, a certain gene and the corresponding mRNA concentration, $y_i$, is assigned. The consideration of only mRNA concentrations in gene regulatory models instead of transcription factor concentrations grounded on the assumption that concentration changes of mRNAs reflect the corresponding protein concentration changes. The rate of mRNA concentration changes are comprised by the functions $f_i \in \mathcal{F}$, which include the regulatory effects of transcription factors and other processes as well as degradation of components. The specific form of the functions, may vary from linear to highly non-linear.

In neural network models the function $f_i \in \mathcal{F}$ capture the combined regulatory effects of all connected genes to gene $i$. Additionally, degradation of the corresponding mRNA is modeled explicitly. The following dynamics are considered to describe gene regulation

$$\frac{\mathrm{d}y_i(t)}{\mathrm{d}t} = a_i S_i(z_i(t)) - d_i y_i(t) \qquad \forall i \in \mathcal{N} \tag{3.2}$$

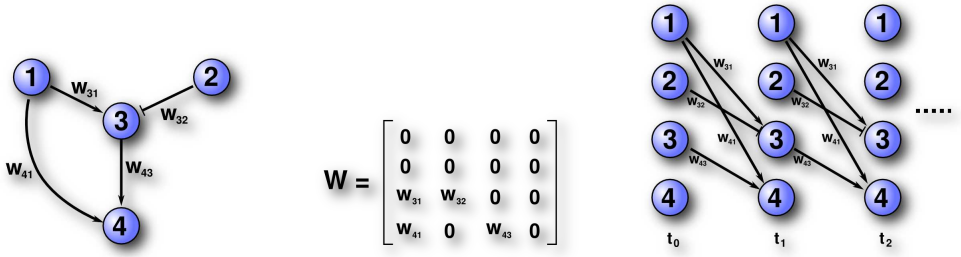
$$\text{with } z_i(t) = \sum_{j \in \mathcal{N}} w_{ij} y_j(t) + b_i. \tag{3.3}$$

Changes of mRNA concentrations are due to synthesis (first term in Eq. (3.2)) and degradation (second term in Eq. (3.2)). The maximal expression rate or activation strength is given by $\mathbf{a} = (a_1, \ldots, a_N)$. Furthermore, degradation is modeled with a linear kinetic, where $\mathbf{d} = (d_1, \ldots, d_N)$ are degradation rates. In contrast, synthesis is described with non-linear kinetics, $S_i$ $\forall i$. Each input signal of connected genes to a particular gene $i$ is weighted by weights $\mathbf{W} = [w_{ij}]_{i,j \in \mathcal{N}}$ and added up. Additionally, bias parameters $\mathbf{b} = (b_1, \ldots, b_N)$ are considered which can be interpreted as reaction delay parameters. Together with the sum of weighted input values they determine the overall input signal, $z_i$. These combined effects are transformed by a sigmoidal activation function,

$$S_i(x) = \frac{1}{1 + e^{-x}}, \tag{3.4}$$

to the interval $(0, 1]$ for $x \in \mathbb{R}$. Sigmoidal functions reflect a saturation effect, i.e., a convergence to a maximal and minimal value for large positive and negative values, respectively. This is assumed to occur in real gene regulation as described in Section 1.1.2. This function type is assumed for all genes in the neural network and, hence, the index $i$ at the activation functions is omitted throughout this thesis. The sigmoidal activation function, Eq. (3.4), is a simplification of transcriptional regulation but is well convenient for learning strategies. However, an alternative activation function is discussed in Section 5.2.

The model parameters, $\mathcal{Q} = \{\mathbf{W}, \mathbf{a}, \mathbf{b}, \mathbf{d}\}$[1], are subject of several constraints; $a_i \in \mathbb{R}_0^+$, $b_i \in \mathbb{R}$, $d_i \in \mathbb{R}_0^+$, and $w_{ij} \in \mathbb{R}\ \forall i, j \in \mathcal{N}$. The neural network model is described in total by $N^2 + 3N$ parameters.




(a) Graph (b) Matrix (c) Unfolding graph

Figure 3.1 | **Neural network representation.** **a** | The topology of a neural network model is represented by a graph. **b** | The weight matrix of the neural network model denotes values associated to edges in the graph. **c** | Graph representation of time-discrete processes obtained by unrolling or unfolding the graph such that each layer (vertical arrangements) represents the system at a certain time point. Such a network has as many layers as time points.

A neural network is graphically represented by a graph similar to the one shown in Figure 3.1a. To each edge a weight is assigned, such that an edge from node $j$ to $i$ has the weight $w_{ij}$. Positive weights are indicated with arrow ending edges and negative weights with bar ending edges in the graph. The topology is as well represented by a weight matrix (Figure 3.1b), where links with $w_{ij} = 0$ do not exist in the graph.

For the description of time-discrete processes with neural networks a discretization of Eq. (3.2) with respect to time is necessary. For each equidistant time point[2], $t \in \{t_0, \ldots, t_m\}$ with $t_i = i \cdot \Delta t$, a discretization gives for a small time distance, $\Delta t$,

$$y_i[t + \Delta t] = y_i[t] + \Delta t \cdot (a_i S(z_i[t + \Delta t]) - d_i y_i[t]) \quad \forall i \in \mathcal{N} \tag{3.5a}$$

$$\text{with } z_i[t + \Delta t] = \sum_{j \in \mathcal{N}} w_{ij} y_j[t] + b_i. \tag{3.5b}$$

A square bracket indicates a continuous value at a discrete time point. It is assumed, that the time step, $\Delta t$, is sufficiently small to capture the changes of the values correctly by this first-order approximation. The substitution $z_i[t + \Delta t]$ denotes the input signal of node $i$ at time point $t + \Delta t$. Eq. (3.5) can be considered as an update rule for all nodes. With the knowledge of all node values at

[1] $\mathbb{R}_0^+$ means $\{x \in \mathbb{R} : x \geq 0\}$.

[2] In general, time differences between subsequent layers do not have to be equal, i.e., $\Delta t[t = t_i] = t_{i+1} - t_i$ can be different for different $t_i \in \{t_1, \ldots, t_m\}$. In case of a time dependent time step the learning algorithm can be adapted (see below).

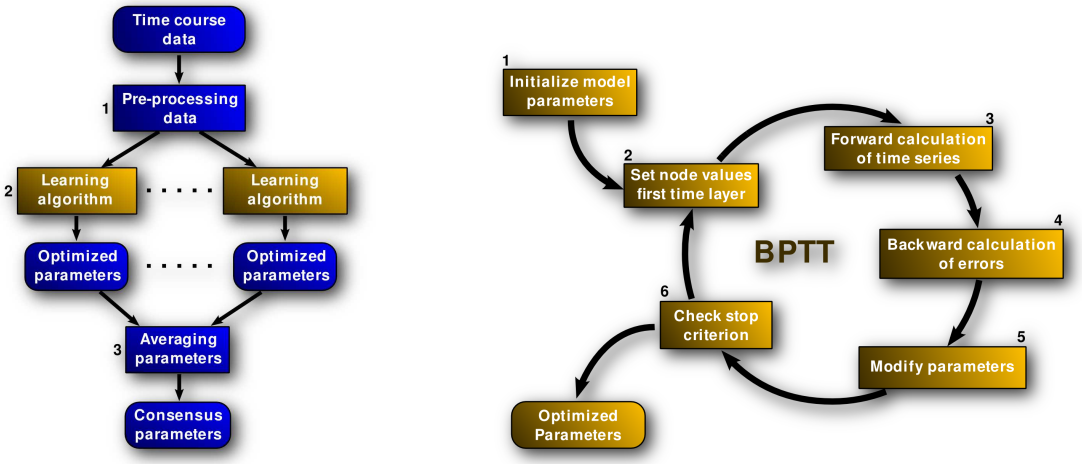
a certain time, the values at the subsequent time point can be calculated. Such temporal processes are Markovian and homogeneous in time, that means, that the state of the system at a certain time point is only dependent on the system state at the previous time point and that the parameters are fixed over time, respectively.

A graphical representation of a time-discrete process is shown in Figure 3.1c. The graph is unfolded with respect to time. Each layer represents the system at a particular time. Connections in the original graph are preserved as links between pairs of subsequent layers. Therefore, the network parameters, $\mathcal{Q}$, are identical in all layers. This concept of unfolding graphs is also used in other dynamical reverse engineering approaches, such as dynamic Bayesian networks.

## 3.2 Learning Algorithm

Based on the model described above a learning strategy can be developed and deployed on a given data set to find an optimal parameter set which explains the data best. The neural network model describes a dynamical gene regulatory system, therefore the data set must contain one or more temporal expression profiles for each gene. These data are the result of various time course measurements where the biological system is perturbed and the relaxation to the steady state is observed. The search for the best parameter set in the parameter space is guided by an adapted decision criterion that evaluates how well the data can be fitted with the current parameter set in the model. Once an optimal parameter set is found the corresponding system can be characterized, e.g., with regard to the network topology.

A supervised learning strategy for adapting the network parameters to temporal data was proposed by Werbos (1974; 1990) and is called backpropagation through time algorithm (BPTT). It was primarily applied to neural networks, econometric models, fuzzy logic structures, and fluid dynamic models with application to pattern recognition, sensitivity analysis, and the control of systems over time, among others. In general, the unfolded network as described in Figure 3.1c is trained with the basic backpropagation algorithm with the restriction that the parameters in all time layers are the same (Werbos, 1990). The BPTT algorithm has the constrained that the model system is represented by functions which are both continuous and differentiable with respect to the values. I have used this strategy to developed a reverse engineering approach called GNRevealer for the analysis of time series data. In the following I describe the general concepts of the BPTT algorithm. More details about the BPTT algorithm are given in Appendix D. of GNRevealer.




(a) GNRevealer Workflow (b) Learning algorithm workflow

Figure 3.2 | **GNRevealer workflow. a** | (1) GNRevealer starts with a pre-processing of the provided time course data, such as interpolation and normalization. (2) Multiple learning procedure (with different initializations) are conducted based on the BPTT algorithm. Each run results in an optimized parameter set. (3) Parameters are averaged to obtain a consensus parameter set as the final result of GNRevealer. **b** | Learning algorithm. (1) The learning procedure starts with a random initialization of all parameter values. Parameters are learned with the BPTT algorithm in an iterative way in step (2)-(6). (2) The node values in the first time layer are set according to the current dataset. (3) Node values of the other time layers are subsequently calculated with the current parameters according to Eq. (3.5). (4) Errors, according to Eq. (3.12), are subsequently calculated from the back through time. (5) All parameters are updated according to Eq. (3.11). (6) Check whether the stop criterion is fulfilled. In that case, the learning process is finished. If not, select randomly a different current dataset and proceed with (2).

The workflow of the GNRevealer program comprises several steps which are shown in Figure 3.2. The program starts with a pre-processing of the provided temporal expression profiles of all genes of interest. These expression values are the result of large scale experiments with DNA arrays and processed in an analysis pipeline described in Section 1.1.3. Usually the data obtained from experiments are noisy and several expression values could not be determined. Therefore, a pre-processing of the data is conducted to obtain smooth time courses by a multivariate Akima interpolation (Akima, 1970) built from piecewise third order polynomials. From the new curves an arbitrary number of interpolated values at time points where changes in the expression of the genes occur are extracted.

If the input data contains large concentration values, very large node values may be estimated during learning, especially in the beginning of the iterations, due to unfitted parameters and, hence, uncontrolled increase of node values over time. In order to avoid such large values during learning the input data

values should be normalized, separately for each dataset. A simple solution would be the division of all values by the overall maximum, $\max_{i,t} y_i[t]$, to map the values into the interval $[0, 1]$. But this might result in different magnitudes of corresponding profiles in different datasets. Instead, a different normalization procedure may be applied. For that, a normalization constant is determined by the median of the values at the last time point. This procedure tries to map the values at the last time point of corresponding profiles to each other, respectively. Besides the value normalization, a normalization should also be applied on the time points by a maximum normalization.

The core of GNRevealer is constituted by the BPTT algorithm. The BPTT learning algorithm is an iterative, gradient based parameter learning method which minimizes the error or cost function

$$E(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{2} \sum_t \sum_i (y_i[t] - \hat{y}_i[t])^2 \tag{3.6}$$

by varying the parameters, $\mathcal{Q} = \{\mathbf{W}, \mathbf{a}, \mathbf{b}, \mathbf{d}\}$, of the model during every iteration step. The first sum extents over all time points and the second over all nodes in the neural network. $\hat{\mathbf{y}} = (\hat{y}_1[t_0], \ldots, \hat{y}_N[t_m])$ are the given expression values of each gene and $\mathbf{y} = (y_1[t_0], \ldots, y_N[t_m])$ are the computed values. The update rules for the node values are given by Eq. (3.5). The error function defined in Eq. (3.6) is proportional to the square of the Euclidean distance between the calculated and the given profiles and is used to asses the quality of the data estimator based on the model and the parameter set.

A stochastic batch learning is selected as the training mode. In a stochastic mode, the parameters of all nodes in the neural network are adjusted in a sequential manner, dataset by dataset. Consequently, estimation of the gradient of the error surface used in the computation is stochastic in nature. Additionally, the sequence of pattern is subject of randomization to avoid artificial oscillations in the parameter modifications.

Another training mode would be batch learning, where the gradient for the update of the parameters is the sum of the gradients caused by each individual dataset. However, batch learning has general inefficiency for gradient descent learning (Wilson and Martinez, 2003). Especially stochastic batch learning shows better performances for large-scale learning of gene regulatory networks from limited data.

In every iteration step of the learning process shown in Figure 3.2b, each parameter $q \in \mathcal{Q}$ is modified according to $q \longrightarrow q + \Delta q$. The modification terms are determined from

$$\Delta q = -\eta_q \frac{\partial^+ E}{\partial q} \tag{3.7}$$

with an ordered derivation operator $\partial^+/\partial q$ which induces a new chain rule

$$\frac{\partial^+}{\partial q} = \frac{\partial}{\partial q} + \sum_{p \in \mathcal{P}} \frac{\partial p}{\partial q} \frac{\partial^+}{\partial p}. \tag{3.8}$$

The sum extents over all variables $p \in \mathcal{P}$ which are explicitly dependent on $q$, i.e., $\partial p/\partial q \neq 0$. This operator is more than a usual derivation with respect to a variable. The usual derivation of a function with respect to a variable considers only the impact of changes of the variable on the function assuming that all other variables are constant. In contrast, the ordered derivation considers as well influences of the respective variable on other variables in the system. With this operator, the following parameter modification term can be derived

$$\Delta q = -\eta_q \sum_{i=1}^{N} \sum_{t=0}^{t_m - \Delta t} \frac{\partial y_i[t + \Delta t]}{\partial q} \delta_i[t + \Delta t], \tag{3.9}$$

with the parameter specific learning rate $\eta_q$ and the error $\delta_i[t]$ at time $t$,

$$\delta_i[t] = \frac{\partial E}{\partial y_i[t]} + \sum_j \frac{\partial y_j[t + \Delta t]}{\partial y_i[t]} \delta_j[t + \Delta t]. \tag{3.10}$$

The first term is the partial derivative of the error function with $\frac{\partial E}{\partial y_i[t]} = y_i[t] - \hat{y}_i[t]$. The second term sums up all errors $d_j[t + \Delta t]$ of all connected nodes to node $i$ weighted by $\partial y_j[t + \Delta t]/\partial y_i[t]$, i.e., the influence of changes in $y_i[t]$ on $y_j[t + \Delta t]$. The error values can only be calculated from the last time layer back through each time layer with the boundary condition $\delta_i[t] = 0$ for $\forall t > t_T$ and $\forall i$. The detailed derivation of the modification and error term is shown in Appendix D. The following parameter modifications have been derived for the BPTT algorithm applied on neural networks

$$\Delta w_{ij} = -\sum_t \eta_{ij}^{(w)} y_j[t] \Delta_i[t + \Delta t] \tag{3.11a}$$

$$\Delta a_i = -\sum_t \Delta t \, \eta_i^{(a)} S_i(z_i[t + \Delta t]) \delta_i[t + \Delta t] \tag{3.11b}$$

$$\Delta b_i = -\sum_t \eta_i^{(b)} \Delta_i[t + \Delta t] \tag{3.11c}$$

$$\Delta d_i = \sum_t \Delta t \, \eta_i^{(d)} y_i[t] \delta_i[t + \Delta t] \tag{3.11d}$$

with

$$\Delta_i[t+\Delta t] = \Delta t\, a_i S_i'(z_i[t+\Delta t])\delta_i[t+\Delta t] \tag{3.12a}$$

$$\delta_i[t] = y_i[t] - \hat{y}_i[t] + \\ + (1-\Delta t\, d_i)\delta_i[t+\Delta t] + \sum_j w_{ji}\Delta_j[t+\Delta t] \tag{3.12b}$$

and the boundary condition $\delta_i[t] = 0$ for $\forall t > t_m$ and $\forall i$. Hence, $\delta_i[t = t_m] = y_i[t] - \hat{y}_i[t]$, $\forall i$.

In case of a time-discrete neural network with varying time steps Eq. (3.11) and Eq. (3.12) can be adapted. For that, $\Delta t$ must be replaced by $\Delta t[t]$ with $\Delta t[t = t_i] = t_{i+1} - t_i$ for $t_i \in \{t_0, \ldots, t_{m-1}\}$ in the respective terms in Eq. (3.11) and Eq. (3.12). However, the time steps need to be small to capture correctly the value changes in the respective time interval by the linear approximation. Too large values may result in neglecting crucial changes in the expression profiles and too small values may result in exceeding computations.

Experimental expression profiles may be incomplete, i.e., not for all genes and all time points a significant expression value can be determined. These are missing values that have to be considered in the reverse engineering method. Either a missing value at a certain time point can be interpolated from temporal adjacent values, e.g., with multivariate Akima interpolation as stated above, or they can be treated separately in GNRevealer. Such a node is considered as hidden and its respective error is calculated according to

$$\delta_i^H[t] = (1-\Delta t\, d_i)\delta_i[t+\Delta t] + \sum_j w_{ji}\Delta_j[t+\Delta t]. \tag{3.13}$$

By changing the parameters during every iteration the estimated values, $\mathbf{y}$, of each node should converge to the given data, $\hat{\mathbf{y}}$, while the parameter modifications, $\Delta q$, decrease with every iteration step. However, usually they do not reach zero, even after many iteration steps. This is due to the stochastic learning mode and incomplete fitting of the data by the estimated neural network model. Therefore, an accurately chosen stop criterion is crucial for a sufficient balance between computation time and accuracy.

In GNRevealer a stop criterion is used (Figure 3.3) which guarantees that all parameter modifications are small over a certain number of iterations. Hence, changes in the values are only expected within small variations for further iterations. For that, an arbitrary number $r \in \mathbb{N}$ and a threshold $R$ is set. $r$ defines the number of values of a parameter $q$, counted from the last iteration step, for that a linear regression, $f(x) = A_q \cdot x + B_q$, is applied. This is performed for

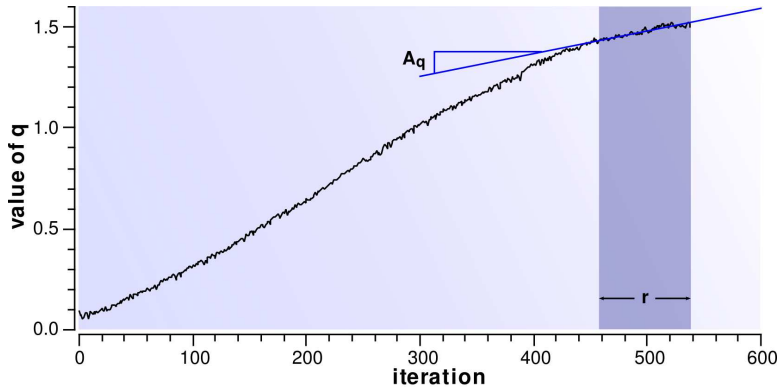



Figure 3.3 | **Stop criterion.** The values of an arbitrary parameter $q$ over iteration steps during parameter learning are shown. For the last $r$ values in the iteration process a linear regression is applied and a slope of the regression line is calculated. The maximal slope of all parameters must be below a threshold to fulfill the stop criterion.

each parameter separately. The stop criterion is then defined by

$$\max_q(|A_q|) < R. \tag{3.14}$$

A similar stop criterion is conceivable where changes in the parameter modification values, $\Delta q$, are observed over iterations. A regression can be similarly applied on a certain number of values. If the maximum of all regression slopes is below a differently defined threshold, the stop criterion is fulfilled. However, the parameters modification errors are dependent on the fluctuations and the correct choice of an appropriate threshold is as well difficult. Furthermore, the parameter modifications are dependent on the learning rate. That is important to consider, if the learning rate is adapted during the learning process.

Once the stop criterion is fulfilled the learning circle is finished and the optimized parameters are processed. For that, the weight matrix is normalized, in such a way that all matrix entries are divided by the maximum, $\max_{i,j}(|w_{ij}|)$. This normalization facilitate the combination of matrix coming from different learning runs and the discretization in a later step.

Gradient descent techniques, such as the backpropagation learning algorithm are known to be limited in finding the global optimum. During search for an optimal solution or global minima, these techniques can encounter local minima from which they cannot escape due to the steepest descent nature of the approach. However, to overcome this difficulty GNRevealer conducts multiple times the learning procedure with the same datasets but with different weight value initializations. Weight values are drawn according to a uniform distribution between $-1$ and 1 or a normal distribution with zero mean and standard deviation of one. Each learning process results in an optimized parameter set.

The learning procedure runs are independent from each other, thus, the computations can be parallelized. This is a further advantage of this approach. To obtain a single result, the parameters are averaged over the runs.

The optimal solution is then given by the averaged learned parameter set where only the weight matrix holds the information about the topology. The other parameters, $\{\mathbf{a}, \mathbf{b}, \mathbf{d}\}$, give additional information about the kinetic but are not further considered. In principle, these parameters can be excluded from learning, but this leads to worse learning results in the weights, since the model with too many fixed parameters is not able to fit the given time courses good enough.

### 3.2.1 Discretization

The output of the GNRevealer is mainly a weight matrix of real numbers. The normalized weights are between $-1$ and 1. Most of the entries in the resulting matrices are usually unequal to zero. Such matrices represent nearly fully connected graphs. In Figure 3.4a the distribution of weight values of many reconstruction results are plotted. In contrast, true artificial networks are sparse which is also assumed for biological networks. Therefore, a discretization of all weight matrix entries into three discrete states (activation, if $w_{ij} \geq \theta$, inhibition, if $w_{ij} \leq -\theta$, and non-regulation, if $-\theta < w_{ij} < \theta$) using a threshold $\theta$ is performed. Since there is no perfect separation of regulations and non-regulations (Figure 3.4a), a discretization threshold has to be chosen accurately.

The performance of a matrix discretization can be evaluated with regard to a target matrix by means of statistical measures. For that, a classification is adapted which considers three states, 1 for activation, -1 for inhibition, and 0 for non-regulation (see Figure 3.5 for the classification matrix). The performance of classification can be statistically measured by sensitivity and specificity. Sensitivity is the fraction of the number of predicted true regulations among all regulations in the reference model. With respect to the ternary classification of values, the sensitivity is defined by

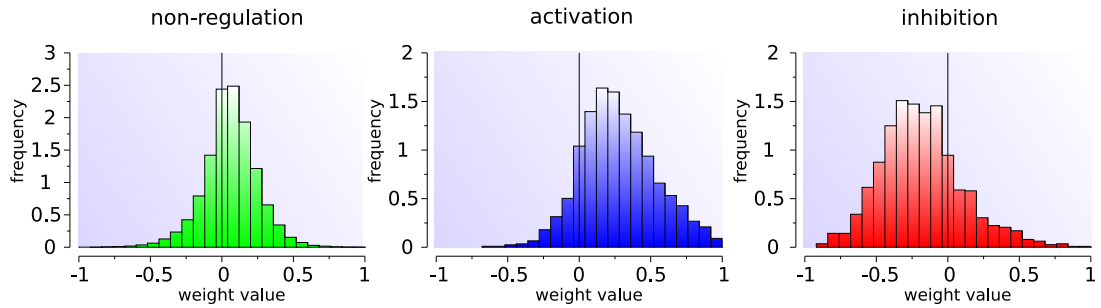
$$\text{sen} := \frac{\text{TR}}{\text{TR} + \text{FZ} + \text{FI}}. \tag{3.15}$$

In contrast, specificity is the fraction of correctly found non-regulations to all non-regulations in the true model and is defined by

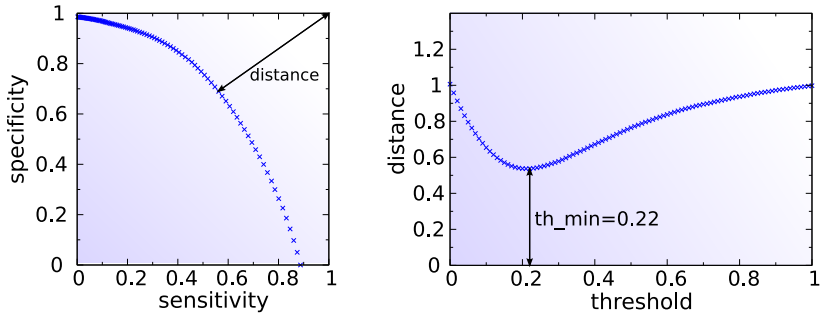
$$\text{spe} := \frac{\text{TZ}}{\text{TZ} + \text{FR}}. \tag{3.16}$$

Additionally, the precision, defined by

$$\text{ppr} := \frac{\text{TR}}{\text{TR} + \text{FR} + \text{FI}}, \tag{3.17}$$




(a) Normalized histograms




(b) ROC and performance over threshold value.

Figure 3.4 | **Determination of an optimal threshold for discretization. a** | Normalized histograms of over 2 700 learned weight matrices as the output of different GNRevealer runs. Datasets of various sizes and noise levels were generated within GeNGe from models with different complexities featuring several biological characteristics. Weights are separated into non-regulations, activations, and inhibitions according to the input models. **b** | *Left:* Plot of specificity (Eq. (3.16)) over sensitivity (Eq. (3.15)). Also shown is the distance $d$ (Eq. (3.18)). *Right:* Plot of $d$ over $\theta$. For $\theta = 0.22$, $d$ is minimal.

is the fraction of true regulations among all found regulations. In Appendix E on page 131, classifications and its statistical measures are visualized.

Sensitivity and specificity can be combined into a single measure, called distance, $d$,

$$d(\text{sen}, \text{spe}) = \sqrt{(1 - \text{sen})^2 + (1 - \text{spe})^2}. \tag{3.18}$$

The distance indicates the similarity of a matrix with regard to a reference matrix after discretization. Its value is between zero and one. Smaller values indicate higher similarity. A perfect match is given by $d = 0$.

A threshold for matrix discretization is determined by an optimization procedure with regard to the distance measure, $d$. A large number of reconstruction results are considered for this optimization strategy. The sensitivity and specificity are the averaged values over all reconstruction results. Distances are calculated for different thresholds and the threshold which minimizes the

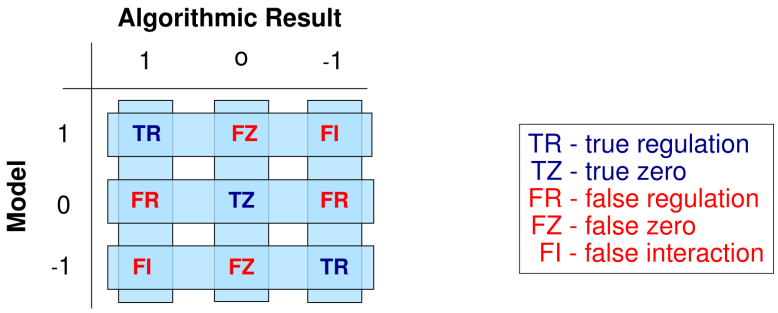



Figure 3.5 | **Classification matrix**. *Left:* Algorithmic results of reverse engineering methods with real-valued output are classified into three states representing activation (1), inhibition (-1), and non-regulation (0). *Right:* Several classification terms are introduced.

distance is selected as an optimal solution. The distance is a balance between sensitivity and specificity, i.e., between calculated true regulations and true zeros (non-regulations) among all regulations and non-regulations in the model, respectively. A lower threshold would result in more true regulations but with more false regulations and less true zeros, i.e., the sensitivity is increased while the specificity is decreased. A higher value has the opposite effect.

The distance can be visualized in a receiver operating curve (ROC), where each point refers to an averaged sensitivity/specificity pair after discretization of a large number of matrices for a certain threshold. The distance, $d$, can be interpreted as the Euclidean distance between such a point and the optimal point $(\text{sen}, \text{spe}) = (1, 1)$. This is shown in Figure 3.4b.

For the optimization procedure of a discretization threshold for reverse engineering results the knowledge of the true networks is essential. However, this knowledge is usually not given and, hence, a threshold has to be determined in advance. For that, a large number of artificial networks which reflect biological features with different complexities and, subsequent, various expression profiles with different noise levels has to be generated. Expected features have to be included, such as network size and noise levels. These data are then analyzed by the reverse engineering method. The resulting matrices are used together with corresponding true matrices to determine an optimized discretization threshold. The generation of a multitude of networks and data is crucial in this procedure. For that, the application GeNGe is preeminently convenient for this task. I conducted such a large optimization process for the study in Section 3.5 comprising more than 2 700 learned matrices (Figure 3.4).

Such discretization of continuous values is a general concept and can be applied on matrices coming from diverse reverse engineering methods. A comparison of the original matrices is often not possible, since they differ in the information they provide and, thus, have to be interpreted differently. Some methods result in correlation measures of genes, some calculate conditional

Table 3.1 | **Comparison of different threshold optimizations.** Mean values of classification of reconstruction results after discretization are calculated (see classification matrix in Figure 3.5) using different thresholds obtained by different optimization procedures. Simulated datasets base on developmental gene regulatory network and comprise different numbers of time series and noise levels. Threshold optimization base on minimizing distance measure defined in Eq. (3.19).

| **Distance** | $d_{(1,1)}$ | $d_{(1,2)}$ |
|---|---|---|
| threshold | 0.22 | 0.28 |
| true regulation (TR) | 13.6(4.4) | 11.2(4.7) |
| true zero (TZ) | 195.1(18.1) | 216.3(13.1) |
| false regulation (FR) | 52.9(18.2) | 31.7(13.1) |
| false interaction (FI) | 0.2(0.5) | 0.07(0.28) |
| false zero (FZ) | 10.2(4.3) | 12.8(4.8) |
| sen | 0.57(0.17) | 0.47(0.20) |
| spe | 0.79(0.17) | 0.87(0.05) |

independencies, and others infer regulation strengths. Therefore, a topology has to be determined from the reverse engineering outputs, e.g., by means of discretization, to enable a comparison of the results.

An alternative distance measure for threshold optimization can be constructed. In Eq. (3.18), the sensitivity and specificity have the same influence on the measure. For a shift to higher sensitivity or higher specificity, different weights, $s_1, s_2$, can be introduced, such that sensitivity and specificity is differently weighted,

$$d_{(s_1,s_2)}(\text{sen}, \text{spe}) = \sqrt{s_1(1-\text{sen})^2 + s_2(1-\text{spe})^2}. \tag{3.19}$$

In order to reduce false regulations, i.e., increasing specificity, at the cost of decreasing sensitivity, weights with $s_2 > s_1$ have to be chosen, e.g., $s_1 = 1$ and $s_2 = 2$. This results in significantly lower false regulations (see Table 3.1) while slightly less true regulations could be identified. However, there is no optimum in the choice of $s_1$ and $s_2$.

## 3.3 Performance

For performance tests, I have implemented parts of the developmental network in sea urchin described by Davidson et al. (2002) (see also Section 2.2.1). The complexity of the networks is characteristic for gene networks, therefore it is used here. I have implemented two models of gene regulation shown in

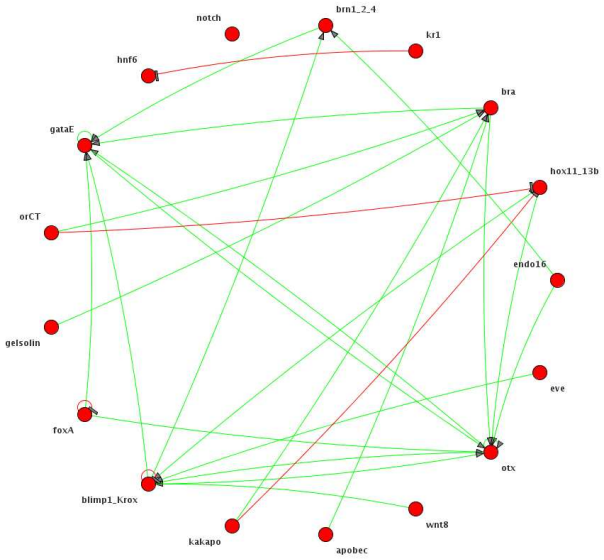
Figure 3.6 using different resources. First, I have used PyBioS[3] for the implementation of a gene regulatory model with 17 genes and mRNAs, 19 proteins, and 4 complexes. They have 28 regulatory interactions (approximately 10% of all possible direct regulations). The PyBioS network graph is too large to be shown here, instead, a small representative is given in Figure 3.6a, where only regulatory interactions are shown. Furthermore, I have used GeNGe for the implementation of a larger part of the developmental network with 25 nodes and 53 regulatory interactions (approximately 9% of all possible direct regulations). For simplicity, protein-protein interactions are not included in the models. In this model protein-protein interactions are not included due to simplicity. Only regulatory interactions are considered.

In a first test case for the performance of GNRevealer several data were analyzed which were generated with a neural network. For the underlying topological structure of the gene regulatory network the 25 node developmental network (3.6c) is selected as a real biological network. The transcriptional kinetic is given in Eq. (3.5). Weights, $\mathbf{W}$, are drawn from a normal distribution around zero and a standard deviation of one. In Figure 3.7 reconstruction results of GNRevealer on data with different sizes are plotted. Obviously, better reconstruction results are obtained for larger datasets. This is indicated in the weight distribution plots by a separation of values corresponding to regulations (blue for activation, red for inhibition) from those of non-regulations. Furthermore, the correlation between true and learned weight values increases with the size of the dataset.

As stated above, a reconstruction result output of GNRevealer is an averaged learned parameter set over multiple repetitions of the learning process. Fluctuations occur due to random initialization of parameters and random selection of the current dataset during learning iterations (Figure 3.2). Therefore, standard deviations of each averaged value can be calculated. These standard deviations represent robustness of the learning process where lower values indicate more stable results with regard to random parameter initializations. These fluctuations are dependent on the dataset size (Figure 3.8). An increase of provided datasets as input for GNRevealer improves not only the correct identification of regulations but also decreases the standard deviations and leads to more trustful results.

A further investigation of the reconstruction results with the developmental network (17 nodes) revealed that several regulations could not be correctly identified even with large input datasets. This can be observed in Figure 3.8. The corresponding weight values (blue color) remain below the discretization threshold while most other values corresponding to regulations could be

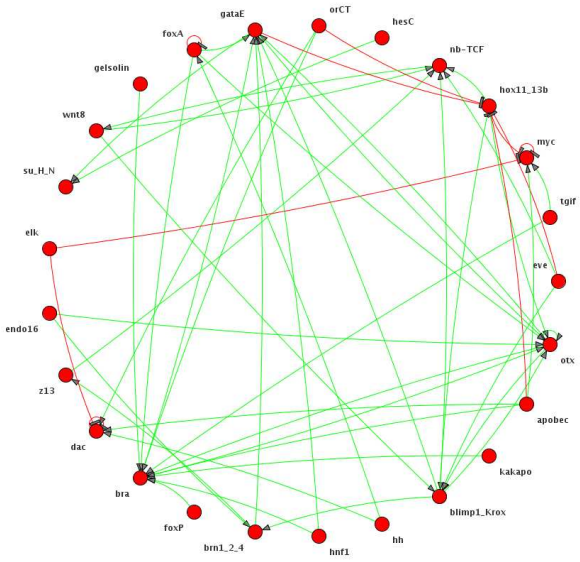
[3] http://pybios.molgen.mpg.de




(a) 17 nodes network

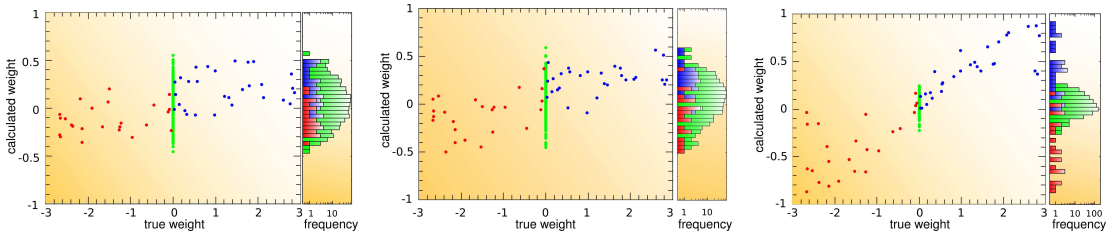
$$\begin{aligned} \text{Notch} + \text{SU(H)} &\leftrightarrows \text{SU(H):N}^{\text{IC}} \\ \text{GSK3}\beta_i + \text{Wnt8} &\leftrightarrows \text{GSK3}\beta_a + \text{Wnt8} \\ \text{GSK3}\beta_a + \beta\text{Catenin} &\rightarrow \text{GSK3}\beta_a \\ \beta\text{Catenin} &\rightarrow \text{n}\beta\text{-TCF} \end{aligned}$$

(b) additional reactions




(c) 25 nodes network

Figure 3.6 | **Developmental networks.** Two versions of the gene regulatory network are used throughout this thesis. Both networks represent active gene regulations in endoderm during early development in sea urchin (18-30 hours) (Davidson et al., 2002). Both networks base on different versions of pictogram given at `http://sugp.caltech.edu/endomes/` **a** | Network with 17 nodes and 28 regulations (23 positives and 5 negatives) represents version Feb. 1st, 2006. **b** | Additional reactions in 17-node network, where SU(H):$N^{IC}$ is transcription factor of *gataE* and nβ-TCF is transcription factor of *wnt8*, *blimp1/Krox*, *hox11/13b*, *eve*, and *kr1*. **c** | Modified network graph. Several components were newly identified as active and several were removed. This network from Oct. 3rd, 2008 consists of 25 genes and altogether 53 regulatory interactions.




(a) 5 time series (b) 10 time series (c) 30 time series

Figure 3.7 | **Reconstruction on data generated with neural network.** GNRevealer is applied on datasets with different sizes (5,10, and 30 time series). See text for model description. Optimized weights are plotted over true weights. Green denotes zero weights (according to the true model), blue denotes positive weights, and red denotes negative weights. Furthermore, calculated weight distributions are shown on the right, respectively.

clearly separated by the learning algorithm for large datasets. This observation is confirmed after several repetitions of the reconstruction process for each dataset size. In Table 3.2 the frequency of a correct identification for each regulation in the network over each reconstruction process is shown, separated for different dataset sizes. Even with large datasets, several regulations could not be reconstructed. According to the underlying gene regulatory network, these regulations are activations and belong to transcriptional regulations of genes where also inhibitors are involved. These inhibitions are modeled with strong inhibitors, i.e., they dominate the regulations and the effect of the activators are concealed. In such a case GNRevealer is not able to identify all regulators due its underlying additive model of gene regulation based on neural networks.

An important criterion for the usability of an algorithm in general is its scalability behavior. The computing time of reverse engineering methods depends strongly on the size of data which is analyzed. In Figure 3.9 the time scaling of GNRevealer is plotted and shows less than a quadratic behavior with the polynomial fit $f(x) \propto x^{1.88}$. Note that the calculation time for one learning process is plotted. However, GNRevealer provides the possibility to repeat the learning process to obtain more robust results. The total computation time for a network of a certain size is then in average the mean computation value times the number of repetitions.

## 3.4 Comparison to Linear Approach

As a further test case of the performance of GNRevealer it is compared to a reverse engineering method based on linear networks. A linear network model of gene regulation is a first-order approximation of the real biological system (D'haeseleer et al., 1999). However, especially for systems near their steady

Table 3.2 | **Reconstruction Results.** Frequency of correctly reconstructed regulations in developmental network (17 nodes) are shown for different dataset sizes (5, 10, 20, 50 time series with 7 time points each). For each dataset size the reconstruction is repeated ten times with different sets of time series. For each regulation, the frequency of the correct identification is shown. Correctly identified regulations in all ten reconstruction runs are highlighted in blue. Regulations which could not be reconstructed at all are highlighted in red.

**5 TS**

| regulation | freq |
|---|---|
| gataE → brn1/2/4 | 0.67 |
| gataE → bra | 0.5 |
| gataE → otx | 0.5 |
| hnf6 → kr1 | 0.33 |
| bra → apobec | 0.33 |
| bra → gelsolin | 0.33 |
| brn1/2/4 → blimp1/Krox | 0.17 |
| brn1/2/4 → endo16 | 0.17 |
| gataE → blimp1/Krox | 0.17 |
| hox11/13b → orCT | 0 |
| hox11/13b → kakapo | 0 |
| otx → hox11/13b | 0 |
| otx → gataE | 0 |
| otx → foxA | 0 |
| otx → endo16 | 0 |
| otx → bra | 0 |
| otx → blimp1/Krox | 0 |
| gataE → foxA | 0 |
| bra → orCT | 0 |
| bra → kakapo | 0 |
| blimp1/Krox → wnt8 | 0 |
| blimp1/Krox → otx | 0 |
| blimp1/Krox → hox11/13b | 0 |
| blimp1/Krox → eve | 0 |

**10 TS**

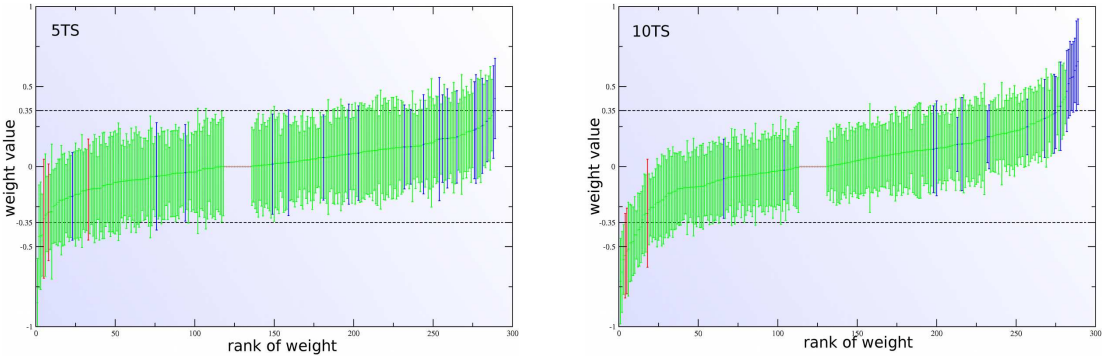
| regulation | freq |
|---|---|
| bra – brn1/2/4 | 1 |
| hox11/13b – orCT | 0.83 |
| bra – gelsolin | 0.83 |
| hox11/13b – kakapo | 0.67 |
| otx – hox11/13b | 0.5 |
| otx – gataE | 0.5 |
| gataE – brn1/2/4 | 0.5 |
| otx – bra | 0.33 |
| otx – blimp1/Krox | 0.33 |
| gataE – bra | 0.33 |
| bra – kakapo | 0.33 |
| blimp1/Krox – wnt8 | 0.33 |
| blimp1/Krox – otx | 0.33 |
| gataE – blimp1/Krox | 0.17 |
| brn1/2/4 – endo16 | 0.17 |
| gataE – otx | 0.17 |
| bra – orCT | 0.17 |
| blimp1/Krox – eve | 0.17 |
| brn1/2/4 – blimp1/Krox | 0 |
| gataE – foxA | 0 |
| otx – endo16 | 0 |
| otx – foxA | 0 |
| hnf6 – kr1 | 0 |
| blimp1/Krox – hox11/13b | 0 |

**20 TS**

| regulation | freq |
|---|---|
| hox11/13b – orCT | 1 |
| hox11/13b – kakapo | 1 |
| otx – hox11/13b | 1 |
| otx – gataE | 1 |
| gataE – brn1/2/4 | 1 |
| gataE – bra | 1 |
| bra – gelsolin | 1 |
| bra – apobec | 1 |
| hnf6 – kr1 | 0.83 |
| gataE – blimp1/Krox | 0.83 |
| gataE – otx | 0.83 |
| otx – bra | 0.83 |
| blimp1/Krox – otx | 0.67 |
| brn1/2/4 – endo16 | 0.67 |
| blimp1/Krox – eve | 0.5 |
| blimp1/Krox – wnt8 | 0.5 |
| otx – blimp1/Krox | 0.5 |
| otx – endo16 | 0.17 |
| bra – kakapo | 0.17 |
| otx – foxA | 0 |
| gataE – foxA | 0 |
| brn1/2/4 – blimp1/Krox | 0 |
| bra – orCT | 0 |
| blimp1/Krox – hox11/13b | 0 |

**50 TS**

| regulation | freq |
|---|---|
| hox11/13b – orCT | 1 |
| hox11/13b – kakapo | 1 |
| hnf6 – kr1 | 1 |
| otx – hox11/13b | 1 |
| otx – gataE | 1 |
| otx – blimp1/Krox | 1 |
| gataE – brn1/2/4 | 1 |
| gataE – blimp1/Krox | 1 |
| bra – gelsolin | 1 |
| bra – apobec | 1 |
| blimp1/Krox – wnt8 | 1 |
| blimp1/Krox – eve | 1 |
| brn1/2/4 – endo16 | 0.83 |
| gataE – bra | 0.83 |
| gataE – otx | 0.83 |
| otx – bra | 0.67 |
| brn1/2/4 – blimp1/Krox | 0.33 |
| otx – endo16 | 0.33 |
| blimp1/Krox – otx | 0.33 |
| bra – orCT | 0 |
| bra – kakapo | 0 |
| gataE – foxA | 0 |
| otx – foxA | 0 |
| blimp1/Krox – hox11/13b | 0 |




(a) 5 time series; sen $= 0.04$, spe $= 0.98$

(b) 10 time series; sen $= 0.46$, spe $= 0.93$

(c) 20 time series; sen $= 0.66$, spe $= 0.98$

(d) 50 time series; sen $= 0.71$, spe $= 0.98$

Figure 3.8 | **Fluctuations of GNRevealer results.** GNRevealer is applied on datasets with different number of time series (5, 10, 20, 50). Data is obtained by simulations with developmental network (17 nodes). The number of time points in each time series is seven. Sorted learned weight values are shown, where green weights correspond to non-regulations, blue to activating regulations, and red to inhibiting regulations according to the source network. Diagonal elements (self-regulations) are set to zero.

state linear models are successfully applicable (Gardner et al., 2003, Bansal et al., 2006) and it is worth to be considered for reverse engineering purposes. A time discrete linear dynamical system is described by

$$\Delta y_i[t + \Delta t] = y_i[t] + \Delta t \cdot \left[ \sum_{j \in \mathcal{N}} w_{ij} y_j[t] + u_i[t] + b_i \right]. \tag{3.20}$$

Note that the weighted input signals, $\sum_j w_{ij} y_j[t]$, are not transferred to a non-linear activation function as done in neural network models. Hence, the rate of changes of a node value remains linearly dependent on each connected node value. The weights, $w_{ij} \in \mathbf{W}$, have the same meaning as in neural networks, i.e., they represent the topological structure of the network. Degradations rates are implicitly contained in the diagonal elements of the weight matrix. Therefore, self-regulatory effects cannot be modeled in linear networks. With the

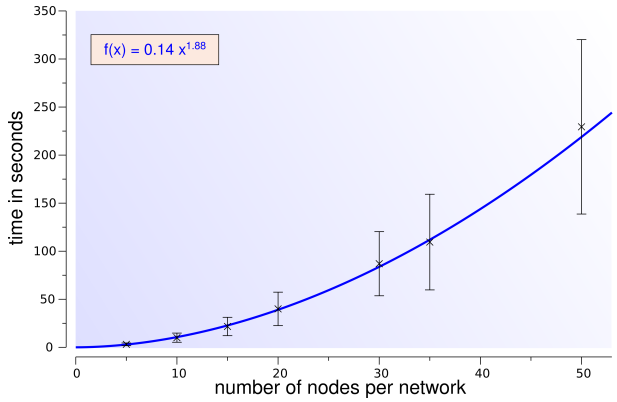
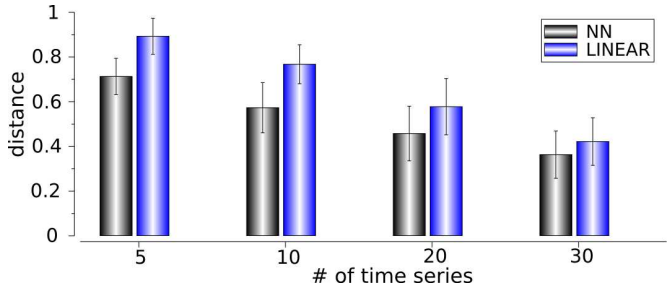
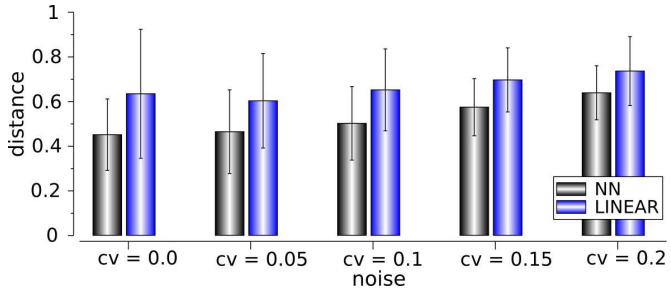



Figure 3.9 | **Scaling behavior of GNRevealer.** Network models of different sizes are generated and time course data are calculated within GeNGe. These data served as input of GNRevealer. Reverse engineering was conducted without repetition of learning procedure. Calculation time of equal data sizes was averaged and plotted.




(a) Distance measure for different dataset sizes




(b) Distance measure for different noise levels

Figure 3.10 | **Comparison GNRevealer and linear approach.** Calculations for the developmental network for different dataset sizes and different noise levels (CV) were done. **a** | For each dataset size (number of time series) the distance measure (Eq. (3.18)) in each result was averaged over all noise levels for each method. **b** | For each noise level the distance was averaged over all dataset sizes.

functions $(u_1[t], \ldots, u_N[t])$ additional external inputs which are not dependent on the node values are included into the model. Furthermore, the parameters, $(b_1, \ldots, b_N)$ realize constant biases in the respective gene expression.

Least square solutions are calculated as described in Hache (2008) using all time series simultaneously to obtain best fitting weight matrices. Following, the learned matrices were discretized using a different pre-determined threshold of $\theta_{\text{opt}}^{\text{LIN}} = 0.26$ which is obtained in a similar threshold optimization procedure. Benchmark data were simulated in GeNGe using a part of the developmental sea urchin network described above as the gene regulatory network with 17 nodes. Gaussian noise were added to the simulated gene expression data. Several datasets of different sizes and different noise levels were collected and analysed by GNRevealer and the linear approach.

For validation of the algorithmic results the distance measure, $d$, which is defined in Eq. (3.18) is used. This performance measure is calculated and plotted in Figure 3.10 for the different datasets. GNRevealer outperforms significantly the linear approach as expected, since with its non-linear model it is more convenient to capture the non-linear dynamics of the input data. With an increasing dataset size the distance decreases in average over different noise levels and with increasing strength of noise the distance increases in average over different dataset sizes. Under each condition GNRevealer gives the better results regarding to the distance measure.

## 3.5 Application to Network Motifs

The reconstruction quality from a dataset depends on the complexity of the underlying network. Figure 3.11 shows the sensitivity and specificity of networks of different motifs in the results. The network models of size fifteen contains the motifs described by Lee et al. (2002), respectively. With increasing noise level of the input data the mean frequencies of correctly identified regulations in a motif (sensitivity) decreases. Each motif shows a different robustness against noise. For perfect data (CV $= 0.0$) the edge detection of a MIM-network is significantly worse than for the other motif networks (66% for MIM and over 74% for SIM, FFL, RC, MCL). But with increasing noise the sensitivity does not decrease as much as the other ones.

In contrast, the specificity increases with larger noise. That is due to averaging of ten reconstruction runs to one result. The same weights of each run differ more with high noise levels but in average these weights tend to remain under the discretization threshold. Therefore the algorithm finds less false regulations (FR) and causes in higher specificity.

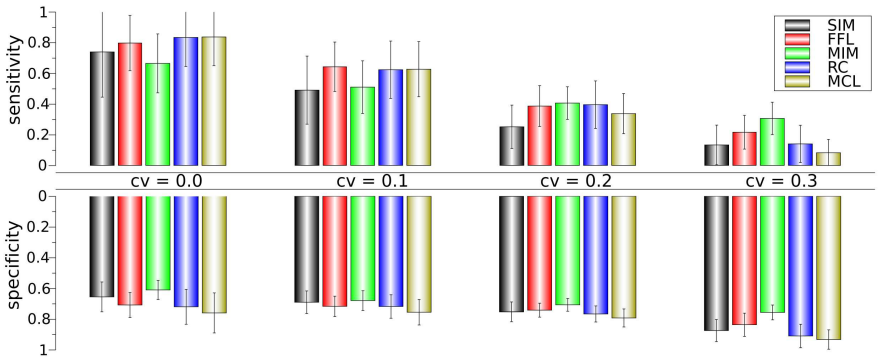



Figure 3.11 | Sensitivity (top) and specificity (bottom) are shown for each motif. The underlying gene networks of size fifteen consist of the small motifs SIM, single input motif, FFL, feed forward loop, MIM, multiple-input motif, RC, regular chain, MCL, multiple-component loop, respectively (Lee et al., 2002). The datasets (each has ten time series) are divided into groups of different noise levels (CV).

## 3.6 Extensions

A combination of neural network learning techniques with Bayes' theorem is a promising approach to improve the learning capacities (Lampinen and Vehtari, 2001). According to Bayes' theorem the posterior probability of a hypothesis, i.e., after observing evidences, is expressed in terms of the conditional probability of seeing the evidences given the hypothesis and the data prior probabilities of the hypothesis and the evidences. Applied this principle to learning techniques for network parameters, $\mathcal{Q}$, with data, $\mathcal{D}$, yields

$$P(\mathcal{Q}|\mathcal{D}) = \frac{P(\mathcal{D}|\mathcal{Q})P(\mathcal{Q})}{P(\mathcal{D})} \tag{3.21}$$

$$\propto P(\mathcal{D}|\mathcal{Q})P(\mathcal{Q}). \tag{3.22}$$

Finding the maximum of the posterior distribution is equal to minimizing the negative logarithm thereof,

$$-\log(P(\mathcal{Q}|\mathcal{D})) \propto -\log(P(\mathcal{D}|\mathcal{Q})) - \log(P(\mathcal{Q})). \tag{3.23}$$

An error or cost function that measures not only the performance of the model with respect to data but also penalize network complexities can be defined as a proportional of this negative log-posterior,

$$E' = -\log P(\mathcal{Q}|\mathcal{D}) - \log(P(\mathcal{Q})), \tag{3.24}$$

where the first term is the performance measure. With a Gaussian assumption for the marginal likelihood, i.e., $P(X_i[t]|\mathcal{Q}) \propto \exp[(y_i[t] - \hat{y}_i[t])^2/2\sigma^2]$ for

independent, identical distributed variables $X_i \; \forall i \in \mathcal{N}$ and a uniform parameter prior, minimization of $-\log P(\mathcal{Q}|\mathcal{D})$ is equal to minimization of the error function, $E$, in Eq. (3.6). A Gaussian prior for all weights[4],

$$P(w_{ij}) \propto P'(w_{ij}) = \exp\left[-\frac{w_{ij}^2}{2\sigma^2}\right], \tag{3.25}$$

yields to a modified error function, $E' = E + \alpha E'_w$, with a non-negative real-valued constant, $\alpha$, and an additional weight decay term,

$$E'_w = \sum_{i,j\in\mathcal{N}} w_{ij}^2. \tag{3.26}$$

This weight decay realizes a certain pressure on all weights during learning to smaller values, especially for large weights. A different parameter prior is proposed by Weigend et al. (1990),

$$P(w_{ij}) \propto P''(w_{ij}) = \exp\left[-\lambda\frac{w_{ij}^2}{w_0^2 + w_{ij}^2}\right], \tag{3.27}$$

that yields a different error function $E'' = E + E''_w$

$$E''_w = \lambda \sum_{i,j\in\mathcal{N}} \frac{w_{ij}^2}{w_0^2 + w_{ij}^2}. \tag{3.28}$$

$\lambda$ and $w_0$ are hyperparameters regulating the additional pressure. It is obvious that $w_0 \gg |w_{ij}| \; \forall i,j \in \mathcal{N}$ yields $E' = w_0^2 E''/\lambda$ and $w_0 \ll |w_{ij}| \; \forall i,j \in \mathcal{N}$ yields $E'' = \lambda N^2$. In contrast, $E'_w$ increases unlimited with increasing $w_{ij}$ (Figure 3.12b).

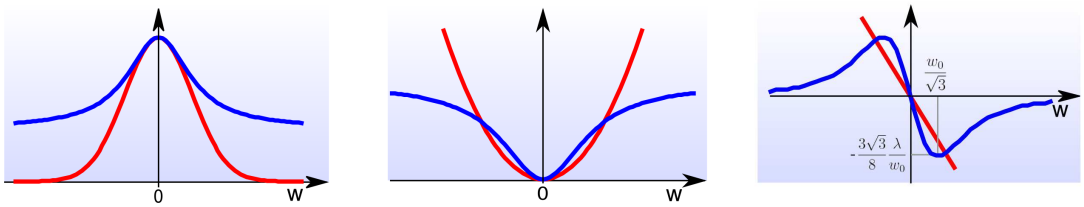
Once a new error function, $E_{\text{mod}} = E + E_w$, can be established the parameter modifications Eq. (3.11) in the BPTT algorithm have to be adapted. For that, the general formalism in Eq. (3.7) has to be applied on the new error function,

$$\Delta q = -\eta_q \frac{\partial^+ E_{\text{mod}}}{\partial q} \tag{3.29}$$

$$\Delta q = -\eta_q \frac{\partial^+ E}{\partial q} - \eta_q \frac{\partial^+ E_w}{\partial q}. \tag{3.30}$$

$E_w$ does not depend on the parameters $\{\mathbf{a}, \mathbf{b}, \mathbf{d}\}$. Therefore, $\partial^+ E_w/\partial q$ is for those parameters zero and their modification terms remain unchanged. For the

[4]It is assumed that $P(\mathbf{W}) = \prod_{i,j} P(w_{ij})$.




(a) $P(w)$ (b) $-\log P(w)$ (c) $-\partial^+ E_w/\partial w$

Figure 3.12 | **Modified errors.** Red curves correspond to Gaussian parameter prior, Eq. (3.25), and blue curves to the prior defined in Eq. (3.27). **a** | The priors are illustrated. **b** | The negative logarithm of the priors. **c** | Additional parameter modification terms due to incorporation of parameter priors (right term in Eq. (3.30)).

weight modifications, $\Delta w_{ij}$, and $E_w = E'_w$ or $E_w = E''_w$, the ordered derivation is equal to the usual derivation, i.e., $\partial^+ E_w/\partial w_{ij} = \partial E_w/\partial w_{ij}$. With this, the following modifications can be derived

$$\frac{\partial E'_w}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}} \sum_{i,i\in\mathcal{N}} w_{ij}^2 \tag{3.31}$$

$$= 2w_{ij}, \tag{3.32}$$

and

$$\frac{\partial E''_w}{\partial w_{ij}} = \lambda \frac{\partial}{\partial w_{ij}} \sum_{i,j\in\mathcal{N}} \frac{w_{ij}^2}{w_0^2 + w_{ij}^2} \tag{3.33}$$

$$= 2\lambda \frac{w_{ij} w_0^2}{(w_0^2 + w_{ij}^2)^2}. \tag{3.34}$$

These modification terms are illustrated in Figure 3.12. The additional values have always the opposite sign compared to the weight values, i.e., they realize an additional pressure on the weights towards zero. Especially Figure 3.12c shows that pressure on the weight values induced by the Gaussian prior, Eq. (3.25), is unlimited for large values. However, a more specific pressure is needed, where a few large values are allowed. This is realized by the prior defined in Eq. (3.27). By means of non-negative real valued hyperparameters, $\lambda$ and $w_0$, the maximum of highest pressure is adjustable (Figure 3.12). The maxima are at $\hat{w} = \pm w_0/\sqrt{3}$ with the values $\mp 3\sqrt{3}\lambda/8w_0$). The hyperparameters have to be chosen that $2\mu\lambda < w_0^2$ with a learning rate $\mu$, otherwise weight modification might be larger than the weight itself and results in an overshoot[5]. $\lambda$ represents the relative importance of the additional penalty term with respect to the performance term. $w_0$ defines the area of maximum pressure (Figure 3.12c).

[5] An approximation of Eq. (3.34) for small $w_{ij}$ is $2\lambda w_{ij} w_0^2/(w_0^2 + w_{ij}^2)^2 \approx 2\lambda w_{ij}/w_0^2$.

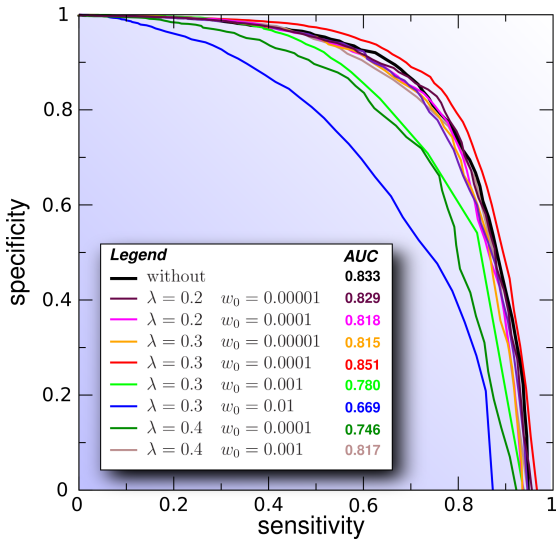



Figure 3.13 | **Reverse Engineering with modified error.** ROC curves are plotted for GNRevealer runs with modified error and different values for the hyperparameters, $\lambda$ and $w_0$. Datasets are based on developmental network (17 nodes, Figure 3.6a). The black curve corresponds to GNRevealer without a modified error. The area under the curves (AUCs) are calculated. Larger values indicate better performances.

I have modified the learning process of GNRevealer with respect to a modified error function. For that, I have considered the parameter prior, $P''(w)$ (Eq. (3.27)) and implemented the modified parameter update rule for weights according to Eq. (3.34). The choice of the hyperparameters, $\lambda$ and $w_0$, is critical for a good reconstruction performance. Unfavorable values may result in significantly worse performances. In Figure 3.13 ROC curves of reverse engineering runs with GNRevealer with different hyperparameter values are plotted. It can be observed that most tested hyperparameter values decreases the performance, i.e., their ROC curves are below the curve of the reconstruction result without a modified error (black line). This is also reflected by the area under the ROC curve (AUC) value, where larger values indicate better performances. A slightly increase in the performance is obtained for a single hyperparameter value pair (red line). However, these results indicate that the correct choice of the hyperparameters is challenging. Even with many pre-simulations, a clear indication for general and significantly better performances could not be derived. Therefore, the modified error approach is not used subsequently.

# CHAPTER 4

# Comparison of Reverse Engineering Methods

Various reverse engineering algorithms have been proposed in order to reconstruct the underlying gene regulatory network from temporal gene expression profiles. But in lack of current experimental time course data it is not clear how these algorithms can be validated. Hence, generating simulated data derived from theoretical considerations is still the method of choice for constructing benchmark data sets. I have conducted a comparative study with six different reverse engineering methods, including relevance networks, neural networks, and Bayesian networks (Hache et al., 2009a). The approach consists of the generation of defined benchmark data, the analysis of these data with the different methods, and the assessment of algorithmic performances by statistical analyses. Performance was judged by network size and noise levels. The results of the comparative study highlight the neural network approach as best performing method among those under study.

## 4.1 Computational Methods

In this study I have selected reverse engineering applications which belong to one of the following classes: relevance networks, graphical Gaussian models, Bayesian networks, or neural networks. In this section I will give an overview

Table 4.1 | **Reverse engineering applications used in this study.**

| Name | Type | Info | Reference |
|---|---|---|---|
| ARACNe (MI) | relevance network with mutual information | C command line | Basso et al. (2005) |
| ParCorA (PC,SC) | relevance network with partial Pearson or Spearman correlation | C command line | de la Fuente et al. (2004) |
| GNRevealer (NN) | neural network | C++ command line | Hache et al. (2007) |
| Banjo (DBN) | Bayesian network | Java command line | Yu et al. (2004) |
| LDST (SSM) | state space model | Matlab script | Rangel et al. (2004) |
| GeneNet (GGM) | graphical Gaussian model | R script | Schäfer and Strimmer (2005a) |

of the basic models, their mathematical background, and discuss the applications I have used. All software can be downloaded or obtained from the algorithm developers. An overview is given in Table 4.1.

### 4.1.1 Relevance Networks

Methods based on relevance networks are statistical approaches that identify dependencies or similarities between genes across their expression profiles. Therefore, relevance networks belong to a statistical reverse engineering approach (Section 1.4). They do not incorporate a specific model of gene regulation. In the first step, the correlation is calculated for each pair of genes, $(X, Y)$, with respect to their vectors of measurement values. Several correlation measures exist, such as mutual information (Shannon and Weaver, 1963)

$$I(X;Y) = H(X) + H(Y) - H(X,Y) \tag{4.1}$$

where $H$ is the Shannon entropy[1], Pearson correlation

$$r_{XY} = \frac{\mathrm{Cov}(X,Y)}{\sqrt{\mathrm{Var}(X)} \cdot \sqrt{\mathrm{Var}(Y)}}, \tag{4.2}$$

and Spearman correlation

$$r^S_{XY} = \frac{\mathrm{Cov}(\mathrm{rg}(X), \mathrm{rg}(Y))}{\sqrt{\mathrm{Var}(\mathrm{rg}(X))} \cdot \sqrt{\mathrm{Var}(\mathrm{rg}(Y))}}, \tag{4.3}$$

with the covariance, $\mathrm{Cov}(X,Y)$, between two random variables $X$ and $Y$, the variance, $\mathrm{Var}(X)$, and the rank vector, $\mathrm{rg}(X)$ of a single random variable $X$. The covariance matrix of the system with the variable set $\mathcal{X}$ is defined by $\mathbf{C} = [\mathrm{Cov}(X_i, X_j)]_{i,j \in \mathcal{N}}$. The widely used Pearson correlation indicates the

[1] Shannon entropy of a random variable is defined by $H(X) = -\sum_{z \in \mathcal{Z}} P(X = z) \log P(X = z)$. $\mathcal{Z}$ is a set of all possible values of $X$.

strength of a linear relationship between the genes. In contrast to that, Spearman's rank correlation and mutual information are able to detect non-linear correlations. It is assumed that a non-zero correlation value for all measures implies a biological relationship between the corresponding genes. Nevertheless, the correlations mentioned above do not capture combined effects of several regulators.

The calculated correlation measures of all gene pairs are usually unequal to zero due to noise and cross-links of genes in a gene regulatory network. After removing of all non-significant links between genes from the proposed network regards to a calculated P-value there are still many remaining links. A non-zero correlation between two genes can be a result of a direct interaction, indirect interaction, or regulation of a common regulator. Hence, direct and indirect interactions have to be identified by the algorithm. Therefore, in a second step of a relevance network approach a pruning process is applied, where the algorithm seeks to remove edges that refer to indirect interactions.

The application Algorithm for the Reconstruction of Accurate Cellular Networks (ARACNe) developed by Basso et al. (2005) uses the Data Processing Inequality (DPI) for that purpose. In each triplet of fully connected nodes in the network obtained after the first step, the edges with the lowest mutual information will be removed. In contrast, de la Fuente et al. (2004) used partial correlations in their proposed method to eliminate indirect interactions. A partial correlation coefficient measures the correlation between two genes conditioning on one or several other genes. In a geometric view, the partial correlation between two variables $X$ with value vector $\mathbf{x}$ and $Y$ with value vector $\mathbf{y}$ conditioning of a set of variables $\mathbf{Z} = \{Z_1, \ldots\}$ is the correlation between the residuals of the projections of $\mathbf{x}$ and $\mathbf{y}$ on the linear space spanned by $\mathbf{Z}$ (Baba et al., 2004). In other words, the residuals result from a linear regression on $\mathbf{Z}$. In general, the partial correlation, $r_{XY.\mathbf{Z}}$, is defined as

$$r_{XY.\mathbf{Z}} = \frac{\mathrm{Cov}(\mathbf{r}_{X.\mathbf{Z}}, \mathbf{r}_{Y.\mathbf{Z}})}{\sqrt{\mathrm{Var}(\mathbf{r}_{Y.\mathbf{Z}})} \cdot \sqrt{\mathrm{Var}(\mathbf{r}_{Y.\mathbf{Z}})}}, \tag{4.4}$$

where $\mathbf{r}_{X.\mathbf{Z}}$ and $\mathbf{r}_{Y.\mathbf{Z}}$ are the residual vectors of $X$ and $Y$, respectively. The residuals represent the respective uncorrelated parts of $X$ and $Y$ with $Z$. The number of genes conditioning the correlation determines the order of the partial correlation.

An inferred network from a relevance network method is undirected by nature. Furthermore, statistical independence of each data sample is assumed, i.e., measurements of gene expression at different time points are assumed to be independent. This assumption ignores the dependencies between time points. Nevertheless, this approach is applied to predict regulatory effects.

I have used ARACNe and the program package ParCorA by de la Fuente et al. (2004) in the comparative study as representatives for relevance network approaches. In the program package ParCorA by de la Fuente et al. (2004) the partial correlations up to 3rd order for Pearson and 2nd order for Spearman correlation are implemented. I have compared all provided correlation measures. Reconstruction results of ARACNe are already discrete information about predicted regulations, since it removes all non-significant links. In contrast, ParCorA results in a real-valued matrix of correlation measures. Therefore, these results are subject of discretization afterwards.

### 4.1.2 Bayesian Networks

A Bayesian network is a stochastic probabilistic graphical network defined by a directed acyclic graph (DAG) which represents the topology and a family of conditional probability distributions. In contrast to other models, the nodes represent random variables and the edges conditional dependence relations between these random variables. Assuming that nodes are only dependent of direct parents (Markov assumption), the joint probability distribution of a Bayesian network can be factorized

$$P(\mathcal{X}) = \prod_{i=1}^{n} P(X_i|\mathcal{X}_{\mathcal{R}_i}). \tag{4.5}$$

For discrete random variables, $P(X_i|\mathcal{X}_{\mathcal{R}_i})$ are multinomial conditional probability distributions. Basically these are tables of probabilities of discrete states for each combination of parents states. With a multinomial distribution nonlinear regulations can be modeled, but the number of free parameters, i.e., entries in all conditional distribution tables is exponential in the number of parents. Therefore, in Bayesian network models often one restricts the maximum number of possible parents.

Continuous node values can be used within a linear Gaussian model which is given by the probability density distribution

$$p(x_i|\mathbf{r}_i) = N\Big(\mu(\mathbf{r}_i) + b_i, \sigma_i^2\Big) \qquad \text{with } \mu(\mathbf{r}_i) = \sum_{j\in\mathcal{R}_i} w_{ij}x_j. \tag{4.6}$$

Each random variable $X_i$ is normally distributed around a mean value $\mu(\mathbf{r}_i)$ which is determined as a sum of weighted parents values. The weights, $w_{ij} \in \mathbb{R}$, are parameters of the model. Due to the linear sum, combinatorial effects of regulators cannot be modeled, such as cooperative binding, only linear relations are considered. To capture nonlinear relationships different mean value

functions, $\mu$, are assumed, e.g. Imoto et al. (2002) used a non-parametric additive regression model based on B-splines to approximate the dependency of the input values.

Static Bayesian networks have several limitations especially for reconstruction purposes. First, several graphs with different edge directions can be consistent with the same joint probability distribution (Chickering, 2002). They are in one equivalence class. That means that they are not distinguishable after learning from data and there is no information of direction for some edges.

Another major drawback of static Bayesian networks is that no cycles are allowed. Since in gene regulatory networks cyclic regulation pathways can occur, dynamic Bayesian network were introduced by Friedman et al. (1998), based on the assumption, that regulations does not happens instantaneous but with a time delay. By unfolding a Bayesian network over discrete time steps (see Figure 3.1c) a valid Bayesian network is again obtained but with a different joint probability distribution

$$P(\mathcal{X}) = P(\mathcal{X}[0]) \prod_t \prod_i P(X_i[t] | \mathcal{X}_{\mathcal{R}_i}[t - \Delta t]), \tag{4.7}$$

where $\mathcal{X} = \{X_1[0], \ldots, X_N[t]\}$ is the extended set of random variables for all nodes and any time $t$. $X_i[t]$ has the value $x_i[t]$. $\mathcal{X}_{\mathcal{R}_i}[t - \Delta t]$ represents the set of parents of node $i$ in previous time slice $t - \Delta t$. The temporal process is Markovian and homogeneous in time, that means a variable $X_i[t]$ depends only on parents in the time slice $t - \Delta t$ and the conditional distribution does not change over time, respectively.

Learning Bayesian networks from a set of measurements, $\mathcal{D}$, means finding a network, $\mathcal{G}^*$, that best matches the given data and the parameters, $\mathcal{Q}^*$, which maximize the posterior parameter distribution given the network $\mathcal{G}^*$. The posterior distribution of network structures and parameters has to be found, given the data and the network and parameters has to be chosen, respectively, which maximize these distributions

$$\mathcal{G}^* = \text{argmax}_{\mathcal{G}}\{P(\mathcal{G}|\mathcal{D})\} \qquad \mathcal{Q}^* = \text{argmax}_{\mathcal{Q}}\{P(\mathcal{Q}|\mathcal{G}^*, \mathcal{D})\}. \tag{4.8}$$

By means of the Bayes rule, the posterior distribution can be written as

$$P(\mathcal{G}|\mathcal{D}) = P(\mathcal{D}|\mathcal{G})P(\mathcal{G})/P(\mathcal{D}) \tag{4.9}$$

$$\propto P(\mathcal{D}|\mathcal{G})P(\mathcal{G}). \tag{4.10}$$

The normalization constant is given by

$$P(\mathcal{D}) = \sum_{\mathcal{G}} P(\mathcal{D}|\mathcal{G})P(\mathcal{G}). \tag{4.11}$$

The marginal likelihood is an integration over the whole parameter space

$$P(\mathcal{D}|\mathcal{G}) = \int P(\mathcal{D}|\mathcal{Q},\mathcal{G})P(\mathcal{Q}|\mathcal{G})d\mathcal{Q}. \tag{4.12}$$

This can be interpreted as an averaging of the probability of generating the data, $\mathcal{D}$, with a graph, $\mathcal{G}$, and parameters, $\mathcal{Q}$, over all possible parameter assignments weighted with the parameter prior, $P(\mathcal{Q}|\mathcal{G})$, of the network. An advantage of using Bayesian models is the possibility of integrating priors for the graphs, $P(\mathcal{G})$, and the parameters, $P(\mathcal{Q}|\mathcal{G})$.

A common approach is to assign a score to networks which evaluates the network with respect to the data. The score is based on the posterior distribution given in Eq. (4.10). Scores based only on the marginal likelihood, Eq. (4.12), are not recommend since complex networks receive higher values which is not desired. To determine a score of a network the high-dimensional integration in Eq. (4.12) has to be computed for each possible graph, but this is usually computationally intractable.

Nevertheless, under certain conditions the integral is analytically solvable. For two function families, $\mathcal{F}$, there are closed forms for the conditional distribution, $P(X_i|\mathcal{X}_{\mathcal{R}_i})$, and parameter priors, $P(\mathcal{Q}|\mathcal{G})$. Multinomial distribution with Dirichlet prior results in the Bayesian Dirichlet equivalent (BDe) score (Heckerman et al., 1995) and linear Gaussian distribution with a normal-Whishart prior results in the Bayesian Gaussian equivalent (BGe) score (Geiger and Heckerman, 1994). If there is no closed form approximations have to be used, such as Bayesian information criterion (BIC) score (Schwarz, 1978). Several other score metrics based on different assumption are proposed, e.g., by Yang and Chang (2002).

Besides the calculation of an appropriate score the search algorithm is as well a crucial point in a Bayesian learning scheme since the number of equivalence classes increases over-exponential in the network size (Gillispie and Perlman, 2001). There are, e.g., hill-climbing algorithm which searches in the graph space the next graph with a higher score by applying local changes to a graph. It is possible that the algorithm runs into a local minimum where it is trapped. Simulating annealing methods have the chance to get out of such a local minimum with a certain probability, which decreases during the process. Another search method is the K2 algorithm developed by Cooper and Herskovits (1992), which is a greedy search algorithm on the parameter space starting with an empty network. But it requires a prior ordering of the nodes as an input from which another network structure will be constructed.

The program package Banjo by Yu et al. (2004), which I have used in this study represents a Bayesian network approach. It follows a heuristic search

approach. It seeks in the network space for the network graph with the best score, based on the BDe metric. The learning algorithm requires discrete values as input. A discretization is performed by the program itself. For that, two methods are provided; interval and quantile discretization. The number of discretization levels can be specified as well. I have used the quantile discretization with five levels. The output network of Banjo is a signed directed graph.

### 4.1.3 Graphical Gaussian Models

Graphical Gaussian models (Schäfer and Strimmer, 2005a, Ma et al., 2007) belong to statistical reverse engineering approaches. They are undirected probabilistic graphical models that allow distinguishing direct from indirect interactions. Graphical Gaussian models behave similar to the widely used Bayesian networks. They provide conditional independence relations between each gene pair. But in contrast to Bayesian networks, graphical Gaussian models do not infer causality of regulations.

Graphical Gaussian models use partial correlation (Eq. (4.4)) conditional on all remaining genes in the network as a measure of conditional independence. Under the assumption of a multivariate normal distribution of the data the partial correlation matrix is related to the inverse of the covariance matrix, $\mathbf{C}^{-1}$ (Edwards, 2000)

$$r_{ij}^{G} = -\frac{C_{ij}^{-1}}{\sqrt{C_{ii}^{-1}C_{jj}^{-1}}}. \tag{4.13}$$

In a reconstruction process of gene regulatory networks the covariance matrix has to be estimated from the given data and, following, to be inverted. From that the partial correlations can be determined according to Eq. (4.13). Afterwards a statistical significance test of each non-zero partial correlation is employed.

A Graphical Gaussian model is implementation GeneNet by Schäfer and Strimmer (2005a). It is a framework for small-sample inference with a novel point estimator of the covariance matrix. An empirical Bayes approach to detect statistically significant edges is applied to the calculated partial correlations.

### 4.1.4 State Space Models

Further reverse engineering approaches are state space models. They constitute a class of dynamic Bayesian networks where it is assumed that the ob-

served measurements depend on some hidden state variables. These hidden variables capture the information of unmeasured variables or effects, such as regulating proteins, excluded genes in the experiments, degradations, external signals, or biological noise.

A basic state space model for gene expression can be represented by a two layer system

$$\mathbf{x}_t = \mathbf{A}\mathbf{x}_{t-1} + \mathbf{w}_t \tag{4.14}$$

$$\mathbf{y}_t = \mathbf{C}\mathbf{x}_t + \mathbf{v}_t \tag{4.15}$$

with a layer of $H$ hidden variables, $\mathcal{X}$, and $N$ observed variables, $\mathcal{Y}$. The state of the observed variables at time $t$ is represented by a $N$-dimensional, real-valued state vector, $\mathbf{y}_t$ and is generated from a $H$-dimensional, real-valued state vector, $\mathbf{x}$, of the hidden variables. The observed variables are associated with measured expression values. Furthermore, it is usually assumed, that the sequence $[\mathbf{x}_1 \cdots \mathbf{x}_T]$ define a first order Markov process. In Eq. (4.14) and Eq. (4.15) $\mathbf{A}$ is denoted as the state dynamic matrix and $\mathbf{C}$ is the observation matrix, representing the influence of the hidden variables on gene expression level at each time point.

An extension to the basic model is proposed by Rangel et al. (2004) which is applied to reverse engineering and used here. The model for gene expression includes crosslinks from the observational layer to the hidden layer

$$\mathbf{x}_t = \mathbf{A}\mathbf{x}_{t-1} + \mathbf{B}\mathbf{y}_{t-1} + \mathbf{w}_t \tag{4.16}$$

$$\mathbf{y}_t = \mathbf{C}\mathbf{x}_t + \mathbf{D}\mathbf{y}_{t-1} + \mathbf{v}_t. \tag{4.17}$$

Matrix $\mathbf{B}$ models additionally the influence of gene expression values from previous time points on the hidden states and matrix $\mathbf{D}$ captures gene-gene expression level influences at consecutive time points. The matrix composition $\mathbf{CB} + \mathbf{D}$ has to be calculated, which captures not only the direct gene-gene interactions but also the regulation through hidden states over time. An non-zero matrix element $[\mathbf{CB} + \mathbf{D}]_{ij}$ denotes activation or inhibition of gene $j$ on gene $i$ depending on its sign.

## 4.2 Validation Measures

For validation of algorithmic results, I calculated the sensitivity, specificity, and precision defined in Eq. (3.15), Eq. (3.16), and Eq. (3.17), respectively, and illustrated in Figure E.1. They are based on a ternary classification (activation, inhibition, non-regulation) of the resulting matrices. This classification requires the knowledge of the true networks. Sensitivity is the fraction of the number

of found true regulations to all regulations in the model. In contrast, specificity defines the fraction of correctly found non-interactions to all non-interactions in the model. Since the number of non-interactions in the model are usually large compared to false regulations, the specificity is hence around one and do not give much information about the quality of the method. Therefore I calculated also precision, which is the fraction of the number of correctly found regulations to all found regulations in the result.

The relevance network and graphical Gaussian approaches give no information about direction of a link. Only undirected graphs can be revealed. Therefore, I used modified definitions for sensitivity, $\text{sen}^{\text{U}}$, specificity, $\text{spe}^{\text{U}}$, and precision, $\text{pre}^{\text{U}}$, that consider a regulation between node $i$ and $j$ in the calculated network as true, if there is a link from node $i$ to $j$ or from $j$ to $i$ in the model network. Furthermore, I calculated a measure which considers an undirected graph and, additionally, does not count false interactions, i.e., false identified activations or inhibitions. The corresponding networks are assumed as undirected with no interactions type, i.e., these are undirected, binary graphs. In that case, Eq. (3.15), Eq. (3.16), and Eq. (3.17) are reduced then to the usual, binary definition of sensitivity and specificity, respectively. The modified measures are denoted with $\text{sen}^{\text{B}}$, $\text{spe}^{\text{B}}$, and $\text{pre}^{\text{B}}$.

To obtain a single value measure for one result I calculated the distance, $d$, defined in Eq. (3.18). This measure combines the sensitivity and specificity equally weighted to a single value measure. Low values indicate good reconstruction performances. Correspondingly to sensitivity and specificity, undirected distance measures are indicated by $d^U$ and binary, undirected measures by $d^B$.

Rather than selecting an arbitrary threshold for discretization the resulting matrices it is also convenient to visualize performances in receiver operator characteristic (ROC) curves (Sonego et al., 2008). ROC curves are graphical plots of sensitivity and specificity or similar pairs, such as precision vs. recall and sensitivity vs. (1-specificity). Each point is a result of a classification with a different threshold. ROC curves are used here to assess method performances and to compare them. To obtain a single value measure one can use the area under the curve (AUC). I calculated AUC of the sensitivity vs. specificity curves as an additional performance measure. Larger values indicate better performances. Note that a value less than 0.5 does not mean anticorrelation, since a random classifier is not here represented by the diagonal due to ternary classification.

## 4.3 Simulation Data

For the comparative study of reverse engineering methods described above I generated a large amount of expression profiles from various gene regulatory networks and different datasets within GeNGe. I performed *in silico* perturbation experiments by varying the initial conditions randomly. In the first step I generated random scale-free networks with different sizes as the underlying structure of the gene regulatory networks. For each generated network a mathematical model of gene regulation is constructed. I selected the transcription function (compare Eq. (2.8)),

$$\phi_n(c_{t_1},\ldots,c_{t_n}) = \begin{cases} \prod_{i=1}^{n}\left[1+(2^{a_{t_i}}-1)\frac{x_{t_i}}{1+x_{t_i}}\right] - \\ \qquad -\prod_{i=1}^{n}\frac{1}{1+x_{t_i}}, & \text{for } n\neq 0, \\ 1, & \text{for } n=0. \end{cases} \tag{4.18}$$

Recall that $c_{t_i} \in \mathbf{c}$ is the concentration and $a_{t_i} \in \mathbf{A}$, the regulation strength assigned to each of the $n$ transcription factors, $t_i \in \mathcal{N}_{R_i}$. The second term in the first case of Eq. (4.18) implements the assumption, that regulated genes does not have a constant production rate. In each generated network the probability that regulation is activating is about 70%, otherwise it is inhibiting. This ratio is arbitrarily chosen, but is motivated by the network proposed by Davidson et al. (2002) (shown in Figure 2.14a), where more activators than inhibitors can be found. The regulation strengths $\mathbf{A}$ are randomly and independently drawn from a uniform distribution over the interval $(0,4)$ and $(0,-4)$ for activators and inhibitors, respectively. All other parameters of the kinetic Eq. (2.1) are set to one, including the enzyme concentrations.

Time series of mRNAs are obtained by first drawing randomly the initial concentrations of each component of the model from a normal distribution with the steady state value of this component as mean and 0.2 as coefficient of variation (CV). Steady states are determined numerically in sufficiently long pre-simulations where changes of concentrations did not anymore occur. The simulations are then performed using the initial conditions. With this approach I simulated global perturbations of the system. I inspected the time series and selected all time series which show similar behavior, i.e., relaxation in the same steady state over time. From the simulated mRNA values I picked 5 values at different time steps during the relaxation of the system as the input data of all reverse engineering algorithms. To simulate experimental errors I added afterwards Gaussian noise with different coefficient of variations (CV) to each expression value independently. The mean of the Gaussian distribution is the unperturbed value and the CV represents the level of noise.

Table 4.2 | **Discretization thresholds for different types of measures.**

| Application | Threshold | Threshold$^U$ | Threshold$^B$ |
|---|---|---|---|
| GNRevealer (neural network) [NN] | 0.14 | 0.18 | 0.16 |
| GeneNet (graphical Gaussian model) [GGM] | - | 0.02 | 0.02 |
| Partial Pearson correlation, 0th order [PC0] | - | 0.15 | 0.10 |
| Partial Pearson correlation, 1st order [PC1] | - | 0.11 | 0.09 |
| Partial Pearson correlation, 2nd order [PC2] | - | 0.09 | 0.06 |
| Partial Pearson correlation, 3rd order [PC3] | - | 0.05 | 0.03 |
| Partial Spearman correlation, 0th order [SC0] | - | 0.10 | 0.10 |
| Partial Spearman correlation, 1st order [SC1] | - | 0.10 | 0.09 |
| Partial Spearman correlation, 2nd order [SC2] | - | 0.07 | 0.06 |

I investigated the impact of different numbers of time series of mRNA concentrations and noise levels on the reconstruction results. For this study I generated randomly five networks of sizes 5, 10, 20, and 30 nodes each. For each network I simulated 5, 10, 20, 30, and 50 time series by repeating the simulation accordingly with different initial values, as described above. For example, for a network of size ten and ten time series, the data matrix contains 500 values (10 nodes $\times$ 10 time series $\times$ 5 time points). I added to the profiles noise with CV equal to 0.0, 0.1, 0.2, 0.3, 0.4, and 0.5. After that I took from each time series five equidistant time points in the region, where changes in the expression profiles occurs. Hence, each reverse engineering application had to analyze 600 datasets (5 $\times$ 4 network sizes $\times$ 5 time series sets $\times$ 6 noise levels).

## 4.4 Performance Results

I have accomplished a systematic evaluation of the performances of six different reverse engineering applications using artificial gene expression data. In the program package ParCorA there are seven correlation measures implemented, including Pearson and Spearman correlation of different orders, which I all used. 600 datasets, with different numbers of genes, dataset sizes, and noise levels, were analyzed by each of the total twelve applications.

For all relevance network methods, graphical Gaussian model, and neural network I determined an optimized threshold for discretization of the results considering all datasets. The procedure of discretization is described in Section 3.2.1. For that, all results are considered. Therefore, there might be a bias to better performances. The calculated thresholds are listed in Table 4.2.

The averaged reconstruction performances over all datasets with regard to different validation measures are given in Table 4.3. Since some applications,

such as relevance networks give no information about the direction of regulation, I calculated as well undirected measures, denoted with $U$. Additionally, I computed measures, which considers undirected results and neglects the kind of interaction information (activation or inhibition). These measures are indicated by B.

None of the reconstruction methods outperforms all other methods. Further, no method is capable of reconstructing the entire true network structure for all data sets. In particular sensitivity and precision are low for all methods. A low precision means that among the predicted regulations, there are only a few true regulations. In the study the precision is always lower than 0.3. This is due to the fact that several input data sets carry a high error level. For example, the input data includes time series with noise up to 50% (CV=0.5). This can bias the performance results. On the other side, the dataset contains small scale time series (5 genes) with up to 50 repetitions and performances are much better with respect to these data (data not shown).

The neural network approach shows the best results among the algorithms tested with regard to the distance measure, $d$, and AUC. On average it identifies over 27% of the directed regulations correctly and 33% without considering the direction, the highest values among all methods. This is remarkable considering the high error level inherent in several data sets. However, simultaneously the specificity is lower than three other methods. That indicates, that many false regulations were identified. Less than 10% of the found regulations are true (precision). However, this is still the best score. In contrast, the Bayesian network approaches, DBN and SSM have a large specificity but with a very low sensitivity. Hence the performances are poor. Only a few regulations were identified and only some of them are true (low precision).

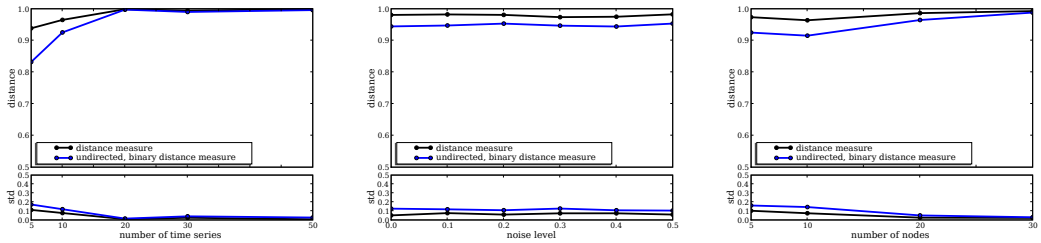
The relevance network approaches using partial Spearman rank correlation show better performances compared to partial Pearson correlation with regard to the distance measure and AUC. This might be explainable by the robustness of the Spearman correlation taking ranks into account rather than actual expression data which is advantageous in spite of noisy data. Surprisingly, with higher orders of partial Pearson and Spearman correlation the distance measures $d^B$ is not increasing. It is around 0.7 for Pearson and 0.68 for Spearman correlation. However, with in average up to 55% ($\text{sen}^\text{B} = 0.545$) of true undirected links could be identified by 1st order Pearson correlation, neglecting the type of interactions. But 0th order Spearman correlation identified over 55% (in average) of all non-regulations.

The MI method (ARACNE) found the fewest true undirected links (low sensitivity $\text{sen}^\text{B}$), except the DBN and SSM methods. In comparison to the relevance network approaches, MI has a considerably larger specificity $\text{spe}^\text{B}$, i.e., MI identifies more non-regulations in the network correctly (true zeros). GGM

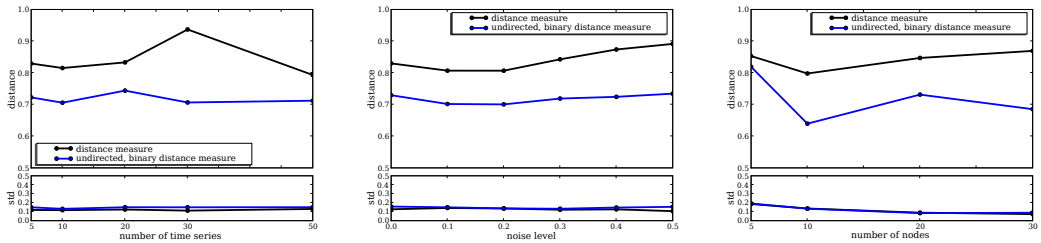
Table 4.3 | **Performance results.** Results of each application applied on all 600 data sets. DBN, Dynamic Bayesian network; NN, neural network; MI, ARACNE (mutual information); PC, partial correlation with Pearson correlation of given order; SC, partial correlation with Spearman correlation of given order; SSM, state space model; GGM, graphical Gaussian model. The column 'type' refers to the performance measure type that can be D for directed graph, U for undirected graph, B for binary and undirected graph. sen, spe, pre, and d are defined in Eq. (3.15), Eq. (3.16), Eq. (3.17), and Eq. (3.18), respectively. AUC is the area under the ROC curve. The averaged values are given with standard deviation in parenthesis. The top value of each type and column is highlighted in boldface. The total number of top values for each method is given in the very last column #TV.

| Name | type | sen | spe | pre | $d$ | AUC | #TV |
|---|---|---|---|---|---|---|---|
| DBN | D | 0.030(0.084) | 0.953(0.117) | 0.041(0.119) | 0.971(0.067) | - | |
| | U | 0.050(0.119) | 0.924(0.173) | 0.064(0.138) | 0.953(0.083) | - | **0** |
| | B | 0.084(0.196) | 0.924(0.173) | 0.099(0.193) | 0.919(0.116) | - | |
| NN | D | **0.276(0.197)** | 0.660(0.216) | **0.091(0.073)** | **0.800(0.131)** | **0.324** | |
| | U | **0.334(0.204)** | 0.617(0.278) | **0.208(0.162)** | **0.768(0.157)** | **0.350** | **11** |
| | B | 0.539(0.255) | 0.574(0.278) | **0.281(0.164)** | **0.628(0.147)** | **0.557** | |
| SSM | D | 0.027(0.073) | **0.973(0.053)** | 0.052(0.139) | 0.973(0.068) | - | |
| | U | 0.030(0.075) | **0.975(0.048)** | 0.094(0.224) | 0.970(0.071) | - | **3** |
| | B | 0.049(0.114) | **0.975(0.048)** | 0.153(0.297) | 0.951(0.110) | - | |
| GGM | D | - | - | - | - | - | |
| | U | 0.238(0.198) | 0.585(0.290) | 0.116(0.157) | 0.868(0.145) | 0.266 | **0** |
| | B | 0.442(0.326) | 0.585(0.290) | 0.225(0.212) | 0.695(0.185) | 0.526 | |
| MI | D | - | - | - | - | - | |
| | U | 0.196(0.129) | 0.745(0.146) | 0.163(0.124) | 0.843(0.130) | - | **0** |
| | B | 0.287(0.177) | 0.745(0.146) | 0.239(0.170) | 0.757(0.162) | - | |
| PC0 | D | - | - | - | - | - | |
| | U | 0.177(0.154) | 0.659(0.234) | 0.106(0.153) | 0.891(0.116) | 0.253 | **0** |
| | B | 0.513(0.228) | 0.492(0.223) | 0.220(0.174) | 0.703(0.132) | 0.506 | |
| PC1 | D | - | - | - | - | - | |
| | U | 0.228(0.161) | 0.541(0.226) | 0.105(0.131) | 0.898(0.126) | 0.249 | **1** |
| | B | **0.545(0.225)** | 0.461(0.214) | 0.219(0.168) | 0.705(0.135) | 0.502 | |
| PC2 | D | - | - | - | - | - | |
| | U | 0.186(0.136) | 0.635(0.154) | 0.108(0.125) | 0.892(0.125) | 0.249 | **0** |
| | B | 0.493(0.186) | 0.515(0.151) | 0.217(0.157) | 0.702(0.140) | 0.504 | |
| PC3 | D | - | - | - | - | - | |
| | U | 0.221(0.154) | 0.573(0.179) | 0.108(0.102) | 0.888(0.135) | 0.250 | **0** |
| | B | 0.526(0.223) | 0.484(0.190) | 0.217(0.142) | 0.701(0.130) | 0.506 | |
| SC0 | D | - | - | - | - | - | |
| | U | 0.285(0.195) | 0.555(0.217) | 0.129(0.135) | 0.842(0.141) | 0.298 | **0** |
| | B | 0.491(0.247) | 0.555(0.217) | 0.230(0.175) | 0.676(0.139) | 0.528 | |
| SC1 | D | - | - | - | - | - | |
| | U | 0.272(0.186) | 0.563(0.215) | 0.127(0.135) | 0.849(0.137) | 0.291 | **0** |
| | B | 0.518(0.233) | 0.518(0.209) | 0.231(0.173) | 0.682(0.135) | 0.521 | |
| SC2 | D | - | - | - | - | - | |
| | U | 0.256(0.165) | 0.588(0.143) | 0.125(0.117) | 0.851(0.137) | 0.290 | **0** |
| | B | 0.500(0.188) | 0.542(0.145) | 0.229(0.155) | 0.678(0.129) | 0.523 | |

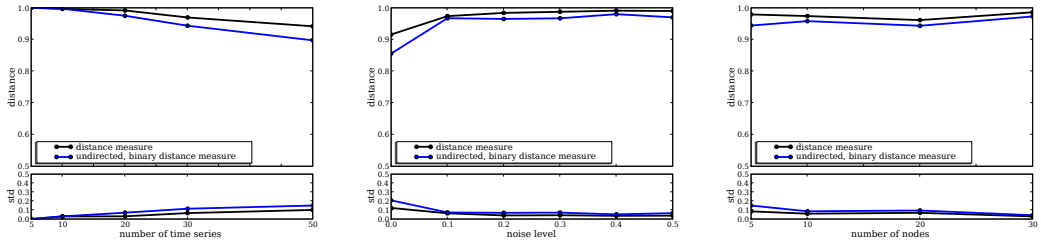
## DBN – Dynamic Bayesian Network (Banjo)




## NN – Neural Network (GNRevealer)




## SSM – State Space Model (LDST)




## GGM – Graphical Gaussian Model (GeneNet)

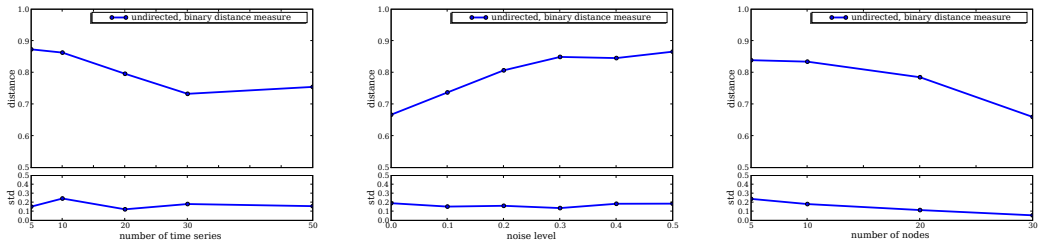
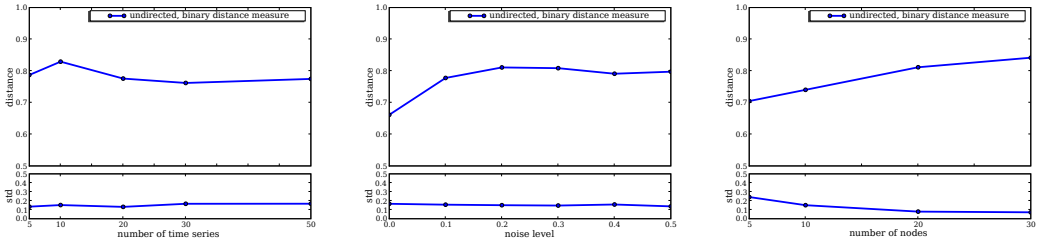
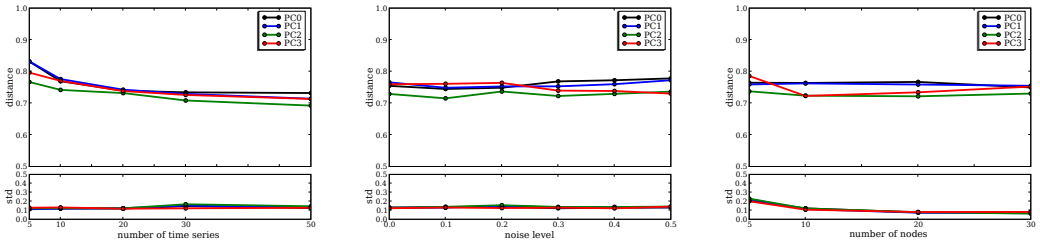



Figure 4.1 | **Performances of applications.** The directed distance measure $d$ (black line) and undirected, binary distance measures $d^B$ (blue line) is plotted with standard deviations below. The measure $d$ is not available for all methods. From left to right in each row: distance measures over number of time series, CV, and network sizes. Each value is an average over all results with the given feature of the abscissa (see text for more details). A smaller distance indicates a better performance.

## MI – Mutual Information (ARACNE)




## PC – Partial Pearson Correlation




## SC – Partial Spearman Correlation

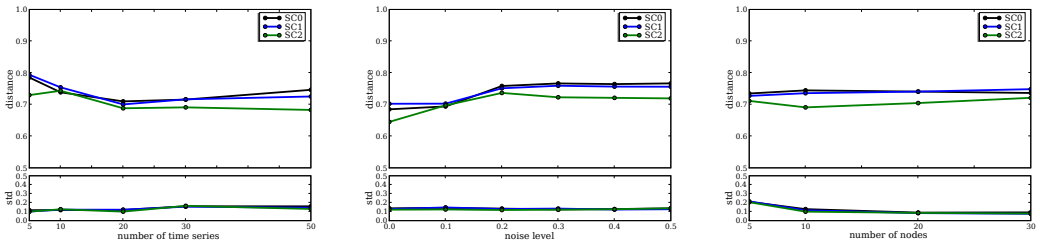



Figure 4.2 | **Performance of applications.** See also Figure 4.1. Only the undirected, binary distance measure $d^B$ is plotted for partial Pearson and partial Spearman correlations. Colors indicate the different correlation measures.

shows the opposite behavior. It has a larger sensitivity but a lower specificity compared to MI.

In Figure 4.1 and Figure 4.2 more details about the performance of each method are plotted with the error resulting from the averaging. The performance behavior with regard to different number of time series, i.e., size of dataset, different noise levels, i.e., coefficient of variation, and network size, i.e., different number of nodes is shown. The distance measures over the number of time series were averaged over five different networks with four different sizes and six different noise levels, i.e., in total of 120 datasets. In case of CV and network size, values were averaged over results from 100 and 150 datasets, respectively.

An overall trend is seen for increasing coefficient of variations. As expected the performances of each method decreases with increasing CV (middle column). Though, the distance measures for Banjo does change only slightly, it remains on a very large value. This indicates a poor reconstruction performance. SSM shows a similar behavior. The distance measure $d^B$ increases very fast for the graphical Gaussian model (GGM). However, the values of the measure decreases noticeable with the size of network. This is in contrast to all other methods, where for larger networks a decrease of reconstruction performance is observable. Surprisingly, the dataset size, i.e., the number of time series does not have a large impact on all methods, except for SSM, where the distance measure decreases from a high value. However, in general, more available data would not always result in a better performance.

Among all partial Pearson correlations methods, the 2nd order outperforms the others. It has a slightly better performance measure under all conditions. This is similar to the 2nd order of partial Spearman correlation. It shows the best performance in all plots. Further, it is always below the best partial Pearson correlation.

## 4.5 Discussion of Performance Results

The comparative study shows that the performances of the reverse engineering methods tested here are still not good enough for practical applications with large networks. Sensitivity, specificity, and precision are always low. Some methods predict only few gene interactions, such as DBN, indicated by a low sensitivity and, in contrast to that, other methods identify many false regulations, such as the correlation measures. I tested different sets of data, including different sizes and noise levels to highlight the conditions for better performances of each method.

DBN performs poorly on all datasets. Under no condition of the study it shows an appropriate performance. The specificity is always very large, but with a very low sensitivity. Only very few regulations were identified and the performance does not improve with larger datasets. It is known that Banjo requires large datasets for better performances (Yu et al., 2004). This may be a reason for the observations. A similar behavior shows the other Bayesian network approach, the state space model. It is slightly better than DBN, but SSM has as well very low sensitivity. The predictive power of such a stochastic approach could not be shown under the conditions in this study.

In contrast, the neural network approach GNRevealer shows the best results among all methods tested. It has a balance between true positives and true zeros. This is due to the appropriately chosen threshold for the post-process discretization. Nevertheless, NN predicts many regulations and many of them are incorrect, i.e., it has many false regulations. Even with a large number of datasets, a complete reconstruction is not possible.

Schäfer and Strimmer (2005a), the authors of the GeneNet package including GGM, pointed out, that their method is intended for analysis of small sample sizes. It requires a large number of genes in the dataset to estimate the null distribution from the data, which is used for detecting statistical significant interactions. This behavior can be observed in Figure 4.1. Increasing number of genes in the data set, decrease the distance measure $d^B$ resulting in a better performance.

The assumption of statistical independence of each time point measurement is not satisfied, although ARACNE performs not all that bad. With larger datasets the performance increases, but decreases, as expected, with noisy data.

The Spearman correlation is a non-parametric measure for correlation. It does not make any assumption for the probability distribution of the data. In this study it outperforms the Pearson correlation, which can only detect linear relationships. It seems, that the rank correlation is more appropriate for analyzing time series data because of its robustness against noisy data.

Crucial in determination of the distance measures are the chosen thresholds for discretization of the resulted matrices of continuous real values. An optimal threshold as shown in Table 4.2 was determined for the methods NN, GGM, all PC, and all SC with regard to an optimized distance measure. The other methods do not need a discretization, since the methods provided a ternary matrix (DBN and SSM) or have already removed all non-significant links form the matrix (MI). A measure, independent of an artificial post-processing of the results is the area under the curve (AUC) score which I calculated as well. The advantage of using AUC is to obviate the need for choosing a discretization threshold. In Table 4.3 it is shown, that a similar classification of method performances is obtain with regard to AUC and the distance measure.

The presented comparative study revealed GNRevealer as the best performing method among those under study. However, all methods have room for improvements. The data requirements on quality and quantity is large and difficult to be provide by experiments. Current reverse engineering methods are not able to reconstruct correctly all gene regulations. But further improvements lead to the right path.

# CHAPTER 5

# Discussion

In this thesis I have addressed several topics of computational analysis of gene regulatory networks. In the first part, I have followed a forward modeling approach. For that, I have developed a new Web application called GeNGe for automatic generation of gene regulatory network models with functionalities for performing various *in silico* experiments. A novel algorithm for the generation of network structures featuring several biological properties has been implemented into GeNGe. Due to the complexity of gene regulation in biological systems, I have established a simplified model for the dynamical description of transcriptional regulation. I have shown that together with a newly derived non-linear transcription kinetic these dynamical models reproduce properties of realistic biological systems. Therefore, GeNGe can be employed for the generation of various *in silico* experiments for predicting effects of perturbations as theoretical counterparts of biological experiments. Investigation of gene regulatory models is supported, e.g., by the topological characterization of given networks. Moreover, GeNGe facilitates especially the collection of benchmark data for evaluating reverse engineering methods.

In the second part of my thesis, I have addressed the reverse problem, the extraction of meaningful information from large amount of data generated by high-throughput genome-wide technologies. Reverse engineering is considered as an equally important challenge as DNA sequencing projects due to the high complexity and dimensionality of gene regulation in biological systems (Tegnér and Björkegren, 2007). I have developed and implemented GNRevealer, a method for reverse engineering of gene regulatory networks from

temporal data. GNRevealer is based on a deterministic neural network model of gene regulation and the known backpropagation through time algorithm. It could be shown in various performance tests using artificial data generated within GeNGe that this method is capable to reconstruct the underlying networks. In a large comparative study, which I have conducted, previously developed reverse engineering algorithms are studied and compared with GNRevealer using data from simulated models. This study draws attention to the necessity for a uniform benchmark that enables an objective comparison and performance evaluation of reverse engineering methods.

## 5.1 Forward Modeling

The advantages in using modeling approaches to gain a comprehensive understanding of a complex system is beyond dispute. Especially, the integration of methods, techniques, and data from different research fields into biology defines a new emergent field called systems biology (Klipp et al., 2005). Systems biology deals with the systematic study of complex interactions in biological systems and offers the chance to predict the outcome of not well understood or unknown processes. Appropriate computational models are developed to explore *in silico* behavior under conditions which might be difficult to realize *in vivo* or *in vitro* with existing experimental techniques. However, models are always a partial representation of reality and capture only certain aspects.

Various levels of accuracy in mapping the reality into computational models have been followed in recent years. Model descriptions are ranging from low detailed models on a high abstraction level to high detailed models with many components representing directly the individual biological counterparts. High detailed modeling requires a multiplicity of information about all individual components, their interactions, and corresponding parameters that need either to be determined experimentally or to be estimated from available data. In contrast, very simplified models may capture general features, however, they may neglect import features of the biological system. For instance, the gene expression is regulated on various levels (Figure 1.1). Each step of gene expression, from the preparation of DNA for transcription to post-translational modifications of proteins, is subject to regulation and modulation by certain factors and may alter the abundances of the final gene products at a certain time and tissue. The most prominent way of gene regulation is the specific binding of transcription factors to DNA regulating the initiation of transcription of polymerase. However, other levels of regulation, such as epigenetic regulation, e.g., DNA methylation and chromatin remodeling, and post-transcriptional regulation by micro RNAs (miRNAs) have turned out to be important to this subject (Mattick, 2004, Weber et al., 2007, Chen and Rajewsky, 2007). However, the

modeling of these additional regulation factors is challenging, since the molecular mechanisms and their effects on transcriptional regulation are partially not well understood. Especially kinetics and their kinetic parameters are essential for detailed modeling.

GeNGe is currently focusing on modeling of gene regulation by transcription factors that is the most prominent part, but might be easily extended with further levels of regulation, since GeNGe uses features of the modeling tool PyBioS via Web services API (see Appendix C). Within PyBioS, very detailed modeling of biochemical networks is supported. This feature can be used for the set-up of models with higher details in GeNGe. However, a primary objective in the development of GeNGe is the nearly automatic generation of realistic gene regulatory models. A general concept for incorporating more details, including more components and appropriate parameter value distributions, has to be developed. Models have to be checked for consistencies, stability, and biological relevance. For instance, unstable systems are biologically unreasonable.

In my thesis I have showed that the novel algorithm for generating gene regulatory network structures allows the design of networks that comprise several features, such as scale-free properties for in- and out-degree distributions and clustering coefficient. Such features are assumed for biological regulatory networks and can be observed in gene regulatory networks extracted from gene regulation databases, such as TRANSFAC (Figure 2.15). The basic principle of the presented network generation algorithm is the random integration of small, biological relevant network motifs. A small set of network motifs, which have been identified in *Saccharomyces cerevisiae* by Lee et al. (2002) and partly in *Escherichia coli* by Shen-Orr et al. (2002), are used as building blocks. However, other network motifs or motif clusters (Kashtan et al., 2004, Koyutürk et al., 2004, Dobrin et al., 2004, Sporns and Kötter, 2004) may be considered in the network generation. Further investigations of biological network structures will reveal the structural and functional properties of frequently occurring subnetwork structures which have to be formulated mathematically. For instance, it would be interesting, if specific subnetworks which belong to the same motif type occur more frequently than others, i.e., what is the probability distribution of all possible subnetwork structures of each motif and, furthermore, what is the probability distribution of the motifs. Considering these properties, the network generation in GeNGe can be refined.

Not only the consideration of relevant components and their composition into interaction networks are important for an appropriate model serving as an image of the biological system but also the type of the underlying mathematical model. Numerous modeling frameworks have been proposed for the description of biological systems (de Jong, 2002, Kaern et al., 2003, Schlitt and Brazma,

2007, Karlebach and Shamir, 2008). Each of them was designed for a certain purpose and is always adapted to the problem domain. There are discrete and continuous, stochastic and deterministic, qualitative and quantitative, spatial and non-spatial models of gene regulation. Each approach has several advantages and dis-advantages. For instance, simplified models are smaller, have only few parameters, and are computationally tractable but capture not all features of the biological system. In contrast, highly detailed models may reflect the biological reality but need for that a large amount of information about individual constituents and their interactions. Moreover, such an approach requires much more computational power for the analysis than for low-detailed ones (Karlebach and Shamir, 2008).

Using discrete models of gene regulation, such as Boolean networks, is justified, if, among others, only limited qualitative information is needed or available. Boolean networks are one of the simplest mathematical models of gene regulation where each node representing a gene is assumed to take one of two values as its state (active or inactive, on or off, expressed or not expressed). Boolean logic describe the regulatory effects between genes. Such models were firstly introduced by Kauffman (1969) to explore dynamic properties of large, randomly constructed gene networks. Although their simplicity, Boolean models can exhibit complex dynamical behavior (Shmulevich et al., 2002). A generalization of Boolean networks are logical networks, where nodes have more than two states (Thomas, 1973).

A Petri Net model is another class of modeling approach for a qualitative description. Petri Nets have been introduced by Petri (1962). The simplest kind of Petri Net is a directed graph consisting of arcs and two different kinds of nodes, place nodes and transition nodes. An arc connects a place with a transition node or vice versa labeled by a positive integer value as a weight. Each place contains tokens. If the number of tokes of each place connected to a transition node larger than the corresponding arc weights, the transition is enabled, to move tokes from the pre-transition to the post-transition place (see Pinney et al. (2003) for a short introduction to Petri Nets). Many extensions were developed, e.g. hybrid, hierarchical, stochastic, and timed Petri Nets. Some are used to model gene regulation, e.g., Goss and Peccoud (1998), Matsuno et al. (2000).

Bayesian network approaches (Pearl, 1988, Friedman et al., 2000), can also be used for forward modeling of gene expression. They are based on a different concept, where nodes represent random variables and edges conditional probabilities. The system is then represented by nodes, edges, and a joint probability distribution. However, different network structures may be represented by the same joint probability distribution, they are equivalent graphs which cannot be distinguished by observation of the variables. Bayesian network models

contain stochastic and qualitative modeling classes. The inherent stochasticity of Bayesian network models takes into account the stochastic aspects of gene regulation and noisy measurements (Kaern et al., 2005). Furthermore, Bayesian networks are either discrete or continuous in the values resulting in different types of conditional probabilities. However, Bayesian networks are usually used as the basis of reverse engineering approaches, as described in Section 4.1.2.

The most prominent way describing gene regulation are differential equations (DEs). DEs allow very detailed description of (dynamical) gene regulatory systems. There are different types of DEs, for instance ordinary differential equations (ODEs), piecewise-linear differential equations (PLDE), partial differential equations (PDE), and qualitative differential equations (QDE) (de Jong, 2002). In ODEs, like the one used in GeNGe, the concentration of all components are modeled by time-dependent variables. Regulatory interactions are described by functions of arbitrary mathematical forms. Therefore, ODEs are the most flexible and widespread modeling technique but requires large efforts for selecting the appropriate functional forms for all actions described in the models.

Finally, the most detailed modeling approach of gene regulation are stochastic simulation algorithms. Especially, when the number of molecules involved in the described processes is low, significant stochastic effects become important and have to be considered (McAdams and Arkin, 1997, Kaern et al., 2005). Recently, single-cell assays demonstrated the stochastic behavior of transcription (Ozbudak et al., 2002, Raj et al., 2006) and translation (Cai et al., 2006, Yu et al., 2006). Therefore, fluctuations have to be considered for single-cell modeling. Stochastic simulations requires a specific mathematical treatment (Gillespie, 1976) and is computationally expensive. Under the assumption that the number of molecules, such as mRNAs and proteins is large, their turnover rates are slow, and many cells contribute to the system, it is convenient to neglect transcriptional and other fluctuations at individual genes and to describe the systems behavior with a deterministic modeling approach, such as ODEs (de Leon and Davidson, 2009). This assumption is also used in GeNGe.

Quackenbush (2001) pointed out that the use of artificially generated data helps to provide an understanding of how data is handled and interpreted by various computational methods, albeit the datasets usually do not reflect the complexity of real biological data. The application to synthetic datasets by computational methods was also proposed by Mendes et al. (2003) for objective comparisons. Furthermore, Repsilber and Kim (2003) followed this approach of using simulated data and presented a framework for testing microarray data analysis tools. Hence, using synthetic data gives the unique possibility for assessing detailed and exact information on performances under well defined

conditions.

Many authors of reverse engineering applications who validated their algorithms with artificially generated data have their own data model. For instance, van Someren et al. (2000) used a linear model generator for their linear approach. As well, Basso et al. (2005) did this for the ARACNe algorithm. Yu et al. (2004) took a stochastic term in a linear data generator model into account in order to test their dynamic Bayesian network method. For the validation of the continuous-time neural network method of Weaver et al. (1999) and Wahde and Hertz (2000), the authors used data produced with neural networks. Finally, the performance test for the REVEAL algorithm of Liang et al. (1998) was performed with a set of Boolean state transitions.

In order to fulfill the above mentioned artificial data requirements and to be independent of the reconstruction method, GeNGe can be used. GeNGe provides an interfaces that is specifically designed to provide an arbitrary number of datasets for benchmarking. Simple as well as complex gene regulatory networks can be generated and simulated.

## 5.2 Reverse Engineering

The knowledge about gene networks and their structure is still limited. For instance, the detailed mechanisms of Hedgehog (HH)/GLI target genes and their associated signaling pathways and regulatory networks in cancer formation are still not well understood. Several investigations indicate that persistent activation of the HH/GLI signal plays a critical role in the initiation and growth of a number of human malignancies including prostate, breast, brain, skin, and pancreatic cancer. Therefore, new strategies, including reverse engineering, have to be developed and applied to obtain a comprehensive understanding of such systems. Time course measurements reveal the behavior of the system under certain treatments and provide data for the identification of the underlying gene regulatory network by reverse engineering strategies. This leads ultimately to an improved and refined model which in turn is used for the generation of further hypotheses by forward modeling approaches which can be tested experimentally. The refinement and validation can be performed in an iterative manner.

Many cellular processes, such as the HH/GLI signal transduction pathway and its downstream gene regulatory network, are dynamic events that can only be detected with time course measurements. Analysis of such data requires mathematical models describing the dynamical behavior. However, it has been shown in this thesis and other studies that reverse engineering is still a challenge for algorithmic developments, especially due to the limited availability

of appropriate experimental data. Despite large-scale measurements, the provided data about single genes are rather small compared to the large number of genes in biological systems. This is known as the dimensionality curse. The space of possible network structures increases super-exponential in the network size (Gillispie and Perlman, 2001) and the dimension of the parameter space is also highly dependent on the network size. As mentioned above, the detailed description of gene regulatory models has to take more parameters into account. Model-based reverse engineering methods may use such models. However, their structure and parameters have to be fitted by sophisticated inference strategies.

GNRevealer performs well in reconstructing networks, but several limitations have turned out, such as low precision and the need for large datasets. However, further extension of GNRevealer are possible. First, the BPTT algorithm is not restricted in the use of the sigmoidal activation function (Eq. (3.4)) in the dynamical model (Eq. (3.2) and Eq. (3.5)). Other functions are also conceivable, however, they have to be at least continuous and differentiable. Therefore, the transcription function that I have derived in Section 2.1.1 can be used for that purpose

$$S_i'(y_1[t],\ldots,y_N[t]) = \prod_{j\in\mathcal{N}} \left[1 + (2^{w_{ij}} - 1)\frac{y_j[t]}{1+y_j[t]}\right]. \tag{5.1}$$

It realizes, like the usual activation function, a saturation for large values. Furthermore, the weights, $w_{ij} \in \mathbf{W}$, represent also regulation strengths. However, the individual regulatory effects of connected nodes are not summed up, like in Eq. (3.4), but multiplied. Therefore, I call it multiplicative model. The use of Eq. (5.1) in this model results in different parameter modification terms. In Section D.2 the new formulae are derived by means of the BPTT formalism.

The proposed multiplicative model capture transcriptional activation better than the additive model, however, the model stability is quite sensitive against random regulation strengths due to the multiplication of the regulation effects, expressed as a product term in Eq. (5.1), of each node. This may result in a large input value and, thus, to a strong activation of a gene for randomly chosen weights. This results finally to large node values over time. The difference between the calculated and given data, expressed in the error function Eq. (3.6), becomes exorbitant large and the next parameter updates may fail. Therefore, small learning rates are required with an appropriate initialization of the weights.

Furthermore, GNRevealer offers the possibility to fix weights and exclude them from learning. For instance, if there is prior knowledge about links which are not present, then the associated weights can be fixed zero. These weights

are excluded from learning such that they will not be updated. Hence, the parameter space is reduced which may result in better convergence of the estimated to the correct data. Moreover, in GNRevealer, certain nodes may be defined as input nodes. That means, that all incoming weights and kinetic parameters associated to that nodes are explicitly excluded from learning, they are not even set fixed. Furthermore, the corresponding expression profiles from data are considered as true, i.e., they will not be calculated. All other parameters are learned given the data. The idea behind this is to improve the learning behavior by define nodes appropriately as transcription factors and find their targets.

Besides large-scale expression data, there are other useful experimental techniques which can be incorporated into reverse engineering methods, such as identifying transcription factor-DNA interactions, measured, e.g., by Chromatin Immuno-Precipitation (ChIP) assays (Ren et al., 2000, Lee et al., 2002), measuring protein expression levels, obtained, e.g., with reverse phase protein assays (Hall et al., 2007), and protein-protein interactions, measures, e.g., with yeast two-hybrid (Fields and Song, 1989). Combining these diverse data could potentially increase the reconstruction capacities. However, the set-up of appropriate models, their learning strategies, and the mathematical formulation of the additional information is challenging.

Even with ideal gene expression data the inference problem is still difficult to solve due to the dimensionality problem (Section 3.3). Further progress in algorithmic development of reverse engineering methods is needed, e.g., by means of biological constraints. The incorporation of biological knowledge mentioned above is in principle possible in neural network models. In Bayesian networks, there is a distinction of network structure and parameters. Therefore, prior knowledge is used for reducing the network space by means of network priors (Imoto et al., 2003, Werhli and Husmeier, 2008). In neural networks, the network structures are induced by weight matrices. Hence, functional forms of parameter priors, $P(\mathcal{Q})$ in Eq. (3.24), have to be developed for integration of prior knowledge. Two types of priors have been discussed already in Section 3.6 which result in modified error functions and, finally, to new parameter update terms. The idea behind these priors is the assumption that most weights (links) are zero and only a few non-zero weights are needed for explaining the data.

Different priors can be constructed by considering knowledge about existing or non-existing links, i.e., regulations in the gene regulatory network. Such information can be retrieved from databases or additional experiments (see above). Parameter priors have to differ between weights, where additional information is available and or not. In the following, four types of information about a specific link are considered: (1) information about activation, (2) infor-

mation about inhibition, (3) information about non-existing regulation, and (4) no information at all. With that, a parameter prior of a particular link weight can be constructed

$$P'''(w_{ij}) = \begin{cases} \exp\left[-\alpha_{ij} w_{ij}^2\right] & \text{for (1)} \\ \left[1 + \exp\left[-\alpha_{ij} w_{ij} - \mu_{ij}\right]\right]^{-1} & \text{for (2)} \\ \left[1 + \exp\left[\alpha_{ij} w_{ij} + \mu_{ij}\right]\right]^{-1} & \text{for (3)} \\ 1 & \text{for (4)} \end{cases} \tag{5.2}$$

with the hyperparameters $\alpha_{ij}$ and $\mu_{ij}$ (see Figure 5.1 for an illustration). Case (1) is a Gaussian distribution with mean zero and standard derivation $1/\sqrt{2\alpha}$ and is already discussed in Section 3.6. Case (2) realizes higher probabilities for larger values and lower probability for lower values. Case (3) has the opposite meaning. Both distributions can be shifted along the x-axis by changing the hyperparameter $\mu_{ij}$. At last, case (4) corresponds to a uniform distribution, i.e., no specific information about the weight values is given. Note that these priors are not probability distributions in a strict sense since they are not normalized. However, normalization is not needed for further derivations.

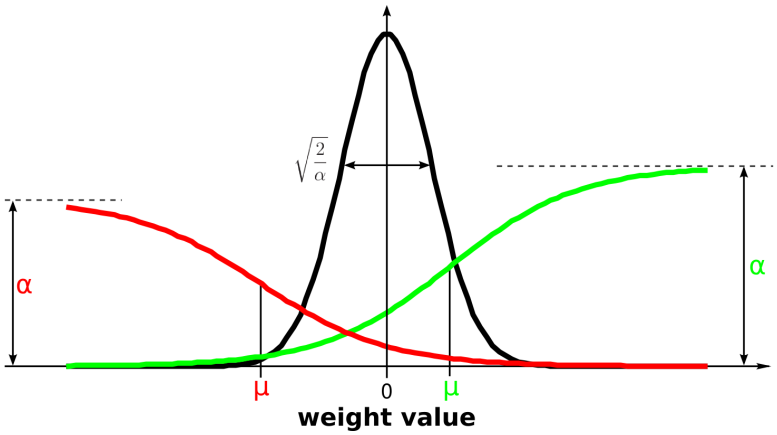



Figure 5.1 | **Proposed parameter prior.** Different cases of parameter priors given in Eq. (5.2) are shown. Black curve corresponds to case (1), green curve corresponds to case (2), and red curve corresponds to case (3). Additionally, the graphical meanings of the hyperparameters, $\alpha$ and $\mu$, are shown.

Weight update rules have to be modified according to the new parameter prior, such that $\Delta w_{ij}^{\text{mod}} = \Delta w_{ij} + \Delta w_{ij}'''$ (according to Eq. (3.30)) with the addi-

tional term $\Delta w'''_{ij} = \partial \log P'''(w_{ij}) / \partial w_{ij}$

$$\frac{\partial}{\partial w_{ij}} \log[P'''(w_{ij})] = \begin{cases} -2\alpha_{ij} w_{ij} & \text{for (1)} \\ \alpha_{ij} \left[1 + \exp\left[\alpha_{ij} w_{ij} + \mu_{ij}\right]\right]^{-1} & \text{for (2)} \\ -\alpha_{ij} \left[1 + \exp\left[-\alpha_{ij} w_{ij} - \mu_{ij}\right]\right]^{-1} & \text{for (3)} \\ 0 & \text{for (4)} \end{cases} \tag{5.3}$$

The hyperparameters have to be estimated from the given biological knowledge. A larger $\alpha$ with $0 \leq \alpha \leq 1$ corresponds to a stronger evidence of the respective case, i.e., information about activation, inhibition, or non-regulation. The value of the hyperparameter $\mu$ is more subtle, since there is no translation from biological knowledge to this value and could be left to a fixed value. However, larger positive $\mu$ values in case (2) and large negative values in case (1) of Eq. (5.3) results in a stronger pressure of weights to larger and lower values, respectively.

## 5.3 Comparative studies

Comparative studies of reverse engineering methods, like the presented one, reveal strength and weaknesses of the individual approaches. In order to perform a comparative study I have chosen six reverse engineering methods proposed in the literature-based on different mathematical models. I was interested in applications that involve the analysis of time series. Furthermore, they should be freely downloadable, easy in use, and having only a few parameters to adjust. Even though there are only a few application parameters which have to be adjusted, each application needs although a special treatment. This includes an establishment of a standardize testing framework. First, provided benchmark data have to be converted into the specific input formats of the individual applications. No standard have been developed for that so far. Since applications are written in different programming languages (C++, R, Matlab, ...), they are invoked differently. However, it is more practicable, especially for a computational parallelization, to have a standardized program call which wrap the actual application. Finally, a post-process of reconstruction results into a comparable form is necessary since application results differ in the information they contain and, thus, have to be interpreted differently. For instance, there are method inferences that result in correlation measures of genes, e.g., relevance networks, some calculate conditional independencies, e.g., Bayesian approaches, and others infer regulation strengths, e.g., neural networks.

A multitude of methods have been developed in recent years to tackle the problem of reverse engineering. However, a comparison of all methods is impossible. The selected methods in this study follow one of the two principle

learning strategies; statistical approach (relevance networks, graphical Gaussian models) and model-driven approach (neural network, Bayesian network, state space models) with different learning algorithms. There are statistical and model-based, directed and undirected, deterministic and probabilistic, discrete and continuous methods. Therefore, the selected approaches cover a wide range of possible learning strategies. Note that a similar comparative study which included a neural network approach has not been performed before.

Some aspects have not been addressed in this study and can be investigated further. It would be interesting to see the performances for larger networks sizes (more than 50 nodes). Some methods, such as GGM should then show increased performances. However, many applications are not suitable for analysing of such large datasets. For these methods a reduction of the dimension of the data has to be performed in order to obtain datasets of appropriate sizes. Different reduction methods can be investigated for that. Moreover, it would be interesting to see the impact of missing data on the reconstruction results, since in real experiments there are often not all genes included in the dataset. Finally, reverse engineering algorithms have to be tested on real biological data. However, as mentioned above, appropriate benchmark data with knowledge of the underlying gene regulatory network is needed. Such data should capture all relevant biological events in the system under study. However, these issues will likely be resolved with technological advancements.

It is shown that the reliable reconstruction of the whole gene regulatory network is any longer an ambitious intention and need further progress. Therefore the quality and quantity of gene expression measurements has to be improved as well as the performance of current or new algorithms. Benchmarks with realistic artificial data has to identify those methods which show the best results under different conditions.

# APPENDIX A

# Network Motifs

Network functions cannot be identified at the level of single genes. Global properties arise from a cooperative action of several genes. Gene regulatory networks contain several substructures that occur frequently throughout many regulatory networks (Shen-Orr et al., 2002, Milo et al., 2002, Lee et al., 2002, Mangan et al., 2003, Odom et al., 2004, Boyer et al., 2005). Such network motifs are topological distinctive interaction patterns composed of only a few nodes and edges. They are considered as the simplest building blocks of complex networks. Studying the functional properties of such motifs may help to understand better global properties of networks (Isaacs et al., 2003). Network motifs do not represent necessarily independent units that are functionally isolated from the rest of the network. Several network motifs interact with each other to aggregate into more complex structures (Dobrin et al., 2004, Ishihara et al., 2005).

In the following several network motifs are introduced which are identified as the simplest building blocks that occur more often in the gene regulatory network of *Saccharomyces cerevisiae* than expected by chance (Lee et al., 2002). I will mention several functional properties and define templates, which are used in a motif search. Such a template is defined as a matrix, where columns and rows represent regulators and targets, respectively. Blue areas define the motif itself. The associated nodes are depicted with M and a number. Consider that motifs usually do not occur isolated from the rest of the network. These other parts of the network are represented as white areas in the template. Corresponding node names start with N. The column entry '1' indicates that the

corresponding link must exist, whereas '0' denotes a link that must not exist in the respective motif. Furthermore, an entry '?' means a link which may exist.

## A.1 Auto-Regulation

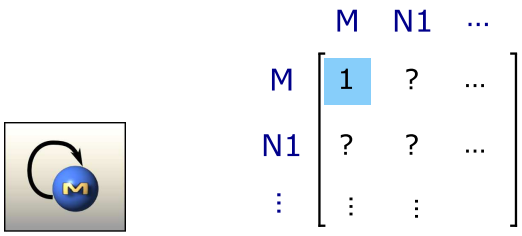



Figure A.1 | Auto-regulation (AR).

Auto-regulation (AR) occurs when a transcription factor regulates the transcription of its own gene. Lee et al. (2002) extrapolated from their studies that about 10% of yeast genes encoding regulators are auto-regulated. There are positive and negative auto-regulators, i.e., activating and repressing regulations, respectively. Negative auto-regulators are considered to speed up the response time of gene circuits and reduce cell-cell variation in protein levels (Alon, 2007). The effect of positive auto-regulators are opposite to those of the negative auto-regulators.

## A.2 Single-Input Motif

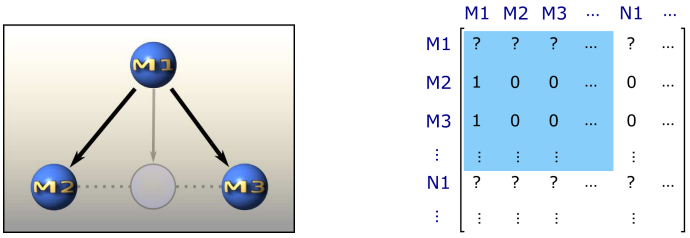



Figure A.2 | Single-input motif (SIM).

Gene regulated exclusively by a single regulator define a single-input motif (SIM). SIMs are coordinating a discrete unit of biological function (Lee et al., 2002). The regulator may generate a temporal order of expression of its target genes (Zaslaver et al., 2004).

## A.3 Feed-Forward Loop

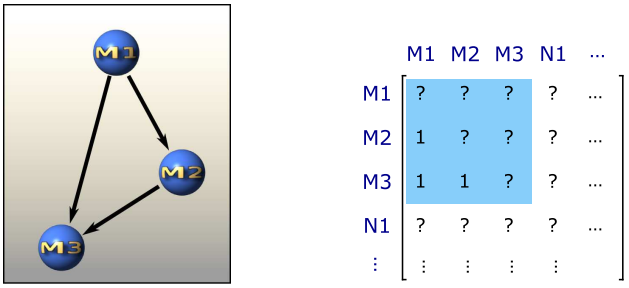



Figure A.3 | Feed-forward loop (FFL).

The feed-forward loop (FFL) motif consists of a regulator regulating a second regulator and both jointly regulating a target gene. This motif is highly favored during evolution. Each arrow represent either an activation of inhibition. However, two out of eight regulation combinations are abundant in gene regulatory networks. In these combinations, all regulations are positive (activating) except the link between $M2$ and $M3$ where it is either positive or negative (inhibiting) in the respective FFL type. If $M1$ is turned on by an external signal, then the transcription of $M3$ is regulated over two branches where the transcriptional signal through $M2$ reaches $M3$ with a time delay and is either activating or inhibiting. FFLs show several features in gene circuits. They provide temporal control, speeding up response time, amplification and introduction of delays of the target gene's expression depending from input signals and the structural types (Lee et al., 2002, Mangan and Alon, 2003, Alon, 2007).

## A.4 Multi-Input Motif

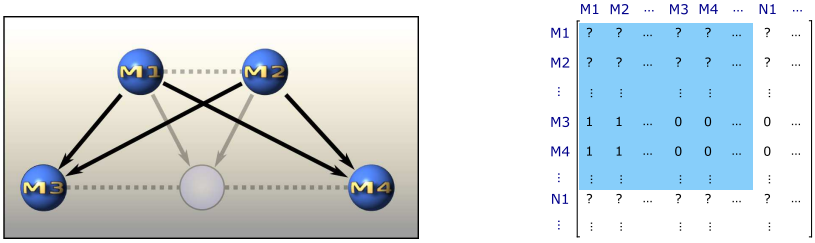
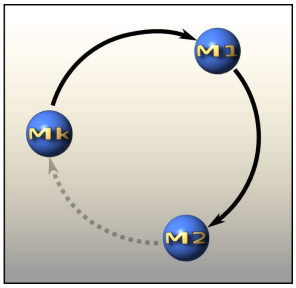



Figure A.4 | Multi-input motif (MIM).

In multi-input motifs (MIMs) a set of regulators are regulating several target genes jointly. This subnetwork is highly densed. Therefore, this motif refer

also to dense overlapping regulons. Such a motif coordinates gene expression across a wide variety of conditions. However, their regulatory input functions are not well understood (Alon, 2007).

## A.5 Multi-Component Loop

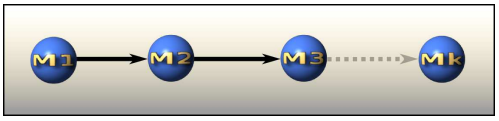



| | M1 | M2 | M3 | ... | Mk | N1 | ... |
|---|---|---|---|---|---|---|---|
| M1 | ? | ? | ? | ... | 1 | ? | ... |
| M2 | 1 | ? | ? | ... | ? | ? | ... |
| M3 | ? | 1 | ? | ... | ? | ? | ... |
| ⋮ | ⋮ | ⋮ | ⋮ | | ⋮ | ⋮ | |
| Mk | ? | ? | ? | ... | ? | ? | ... |
| N1 | ? | ? | ? | ... | ? | ? | ... |
| ⋮ | ⋮ | ⋮ | ⋮ | | ⋮ | ⋮ | |

Figure A.5 | Multi-component loop (MCL).

A multi-component loop is a closed path with at least two nodes in a gene regulatory network ending with the starting regulator. MCL provides the capacity of feedback control and, hence, the establishment of a bistable system.

## A.6 Regular Chain




| | M1 | M2 | M3 | ... | Mk | N1 | ... |
|---|---|---|---|---|---|---|---|
| M1 | ? | ? | ? | ... | 0 | ? | ... |
| M2 | 1 | ? | ? | ... | 0 | ? | ... |
| M3 | ? | 1 | ? | ... | 0 | ? | ... |
| ⋮ | ⋮ | ⋮ | ⋮ | | ⋮ | ⋮ | |
| Mk | ? | ? | ? | ... | 0 | ? | ... |
| N1 | ? | ? | ? | ... | 0 | ? | ... |
| ⋮ | ⋮ | ⋮ | ⋮ | | ⋮ | ⋮ | |

Figure A.6 | Regular Chain (RC).

A regular chain (RC) is simply a set of three or more regulators which regulate each other subsequently arranged in a chain. It realizes temporal ordered transcriptional events.

# APPENDIX B

# Transcription Function

Within GeNGe the bio-logic described by Schilstra and Nehaniv (2008) for transcription kinetics of gene regulatory networks is adapted. In the following the derivation of the provided kinetics is shown.

Starting with an example, the transcription regulation function $\phi_n$ for one gene with three transcription factors (TF) with the concentrations $c_1$, $c_2$, and $c_3$ is given by

$$\begin{aligned}
\phi_3(c_1, c_2, c_3) &= \frac{D}{N} \qquad \text{with} \\
D &= w_{(0,0,0)} + w_{(1,0,0)}x_1 + w_{(0,1,0)}x_2 + w_{(0,0,1)}x_3 + \\
&\quad + w_{(1,1,0)}r_{(1,1,0)}x_1x_2 + w_{(1,0,1)}r_{(1,0,1)}x_1x_3 + \\
&\quad + w_{(0,1,1)}r_{(0,1,1)}x_2x_3 + w_{(1,1,1)}r_{(1,1,1)}x_1x_2x_3 \\
N &= w_{(0,0,0)} + w_{(1,0,0)}x_1 + w_{(0,1,0)}x_2 + w_{(0,0,1)}x_3 + \\
&\quad + w_{(1,1,0)}x_1x_2 + w_{(1,0,1)}x_1x_3 + w_{(0,1,1)}x_2x_3 + \\
&\quad + w_{(1,1,1)}x_1x_2x_3.
\end{aligned} \tag{B.1}$$

The definitions of the parameters is given below. The general definition of the regulation function dependents on an arbitrary number of TFs. The regulation

function reads

$$\phi_n(c_1,\ldots,c_n) = \frac{\sum_{\mathbf{v}} w_{\mathbf{v}} r_{\mathbf{v}} \prod_{i=1}^{n} x_i^{v_i}}{\sum_{\mathbf{v}} r_{\mathbf{v}} \prod_{i=1}^{n} x_i^{v_i}} \qquad \text{with } x_i = \frac{c_i}{K_i}. \tag{B.2}$$

The transcription is stimulated by certain combinatorial effects of TFs with different strengths. Every configuration of $n$ TFs is assigned a binary vector $\mathbf{v} = (v_1,\ldots,v_n)$, where $v_i \in \{0,1\}$ denotes occurrence of the $i$th TF in this configuration. There are $2^n$ possible configurations of TFs (including the one with no TFs, $\mathbf{v} = (0,\ldots,0)$, i.e., unbounded DNA). $\sum_{\mathbf{v}}$ is the sum over all such configurations. $\{w_{\mathbf{v}}\}$ are modulation coefficients which describe the contribution of a TF configuration $\mathbf{v}$ on the transcription rate, with $w_{\mathbf{v}} \geq 0$. $w_{\mathbf{v}} = 0$ means that the configuration $\mathbf{v}$ has no influence on the transcription.

A measure of cooperativity in the binding of a certain configuration $\mathbf{v}$ of regulators is given by the parameter $r_{\mathbf{v}}$. For the unbounded DNA ($\mathbf{v} = (0,\ldots,0)$) and single TFs the cooperative factors are by definition $r_{\mathbf{v}} = 1$. For non-cooperative transcription factor binding I get $r_{\mathbf{v}} = 1$ which I assume in the models. $x_i$ is the normalized TF concentrations $c_i$ normalized by a factor $K_i$. Latter is the equilibrium dissociation constant of the complex DNA $\circ$ TF$_i$. Every gene can have different numbers of transcription factors and modulation coefficients.

To describe joint binding of TFs, i.e., binding of multiple TFs jointly to the promoter of a single gene, the transcription function Eq. (B.2) has to be modified. The TFs in such a set do not effect individually the transcription rate of the target gene. However, if they bind together, they contribute to the transcription rate as an inducer or repressor with a certain regulation strength.

Instead of single TFs I consider $\nu$ sets of joint interactions $I_1,\ldots,I_\nu$ with $\nu \leq n$ (see Figure 2.2). Each set can contain one or more TFs and each TF is a member of exactly one set. Therefore, all sets are disjoint and the conjunction of all sets is equal to the set of all TFs. In the modified transcription function $\phi'$ all configurations of interaction sets are considered instead of configurations of single TFs. Omitting the prime for indication of modified definitions ($\phi' \to \phi$), the transcription function reads

$$\phi_n(c_1,\ldots,c_n) = \frac{\sum_{\mathbf{v}} w_{\mathbf{v}} r_{\mathbf{v}} \prod_{i=1}^{\nu} \prod_{x\in I_i} x^{v_i}}{\sum_{\mathbf{v}} r_{\mathbf{v}} \prod_{i=1}^{\nu} \prod_{x\in I_i} x^{v_i}}. \tag{B.3}$$

Here, $\mathbf{v}$ is a redefined binary vector of length $\nu$ assigned to a configuration, where $v_i \in \{0,1\}$ denotes occurrence of the $i$th interaction set in this config-

uration. $w_{\mathbf{v}}$ is the modulation coefficient and $r_{\mathbf{v}}$ is the cooperativity factor of configuration $\mathbf{v}$. The normalized concentrations were as well redefined. The normalization constant of a certain TF is dependent on the interaction set $I_k$ to which the TF belongs to and is equal to the dissociation constant of the TFs-DNA complex DNA $\circ$ $I_k$, i.e., binding of all TFs in this particular interaction set to the DNA.

With the substitution

$$\tilde{x}_i = \tilde{x}(I_i) = \prod_{x \in I_i} x \tag{B.4}$$

one obtains for Eq. (B.3)

$$\phi_n(c_1, \ldots, c_n) = \frac{\sum_{\mathbf{v}} w_{\mathbf{v}} r_{\mathbf{v}} \prod_{i=1}^{\nu} \tilde{x}^{v_i}}{\sum_{\mathbf{v}} r_{\mathbf{v}} \prod_{i=1}^{\nu} \tilde{x}^{v_i}}. \tag{B.5}$$

Note that the function is still implicitly dependent on all TF concentrations $\{c_i\}$.

For the example given in Figure 2.2, the transcription function of a gene with three transcription factors $TF_1$, $TF_2$, $TF_3$, where $TF_1$ is an inhibitor and $TF_2$ and $TF_3$ are in one interaction set, is given by

$$\begin{aligned} \phi_3(c_1, c_2, c_3) &= \frac{D}{N} \qquad \text{with} \\ D &= w_{(0,0,0)} + w_{(1,0,0)}\tilde{x}_1 + w_{(0,1,1)} r_{(0,1,1)} \tilde{x}_2 \tilde{x}_3 + \\ &\quad + w_{(1,1,1)} r_{(1,1,1)} \tilde{x}_1 \tilde{x}_2 \tilde{x}_3 \\ N &= w_{(0,0,0)} + w_{(1,0,0)}\tilde{x}_1 + w_{(0,1,1)} \tilde{x}_2 \tilde{x}_3 + w_{(1,1,1)} \tilde{x}_1 \tilde{x}_2 \tilde{x}_3. \end{aligned} \tag{B.6}$$

I simplified the transcription function Eq. (B.5) by considering interaction set specific regulation strengths $\{a_i\}$ independent on all other interaction sets, but still specific for the target gene. $a_i > 0$ means activation of the target gene by a factor of $2^{a_i}$ and $a_i < 0$ means inhibition by a factor of $2^{-a_i}$. A modulation coefficient of a configuration is composed by a product of each interaction set specific regulation strength

$$w_{\mathbf{v}} = \prod_{i=1}^{\nu} (2^{a_i})^{v_i}. \tag{B.7}$$

Furthermore, I assume that the interaction sets bind independently from each other, i.e., $r_{\mathbf{v}} = 1 \ \forall \mathbf{v}$. With this assumption, Eq. (B.7), and $w_{\mathbf{v}=(0,\ldots,0)} = 1$ I

obtain for Eq. (B.5)

$$\phi_n(c_1,\ldots,c_n) = \frac{\sum_{\mathbf{v}} \prod_{i=1}^{\nu} [2^{a_i}\tilde{x}_1]^{v_i}}{\sum_{\mathbf{v}} \prod_{i=1}^{\nu} \tilde{x}_1^{v_i}} \tag{B.8}$$

$$= \frac{\prod_{i=1}^{\nu} [1 + 2^{a_i}\tilde{x}]}{\prod_{i=1}^{\nu} [1 + \tilde{x}_1]}. \tag{B.9}$$

The step from Eq. (B.8) to Eq. (B.9) can be proved by mathematical induction for every natural number $\nu$, considering that $\sum_{\mathbf{v}}$ is the sum over all binary vectors of length $\nu$. Eq. (B.9) can also be written as

$$\phi_n(c_1,\ldots,c_n) = \prod_{i=1}^{\nu} \left[1 + (2^{a_i} - 1)\frac{\tilde{x}_i}{1 + \tilde{x}_i}\right]. \tag{B.10}$$

With $w_{\mathbf{v}=(0,\ldots,0)} = 0$, i.e., no basal expression of regulated genes, one obtains

$$\phi_n(c_1,\ldots,c_n) = \frac{\sum_{\mathbf{v}} \prod_{i=1}^{\nu} [2^{a_i}\tilde{x}_i]^{v_i} - 1}{\sum_{\mathbf{v}} \prod_{i=1}^{\nu} \tilde{x}_i^{v_i}} \tag{B.11}$$

$$= \frac{\prod_{i=1}^{\nu} [1 + 2^{a_i}\tilde{x}_i] - 1}{\prod_{i=1}^{\nu} [1 + \tilde{x}_i]}. \tag{B.12}$$

As stated above, the step from Eq. (B.11) to Eq. (B.12) can be proved by mathematical induction for every natural number $\nu$. One obtains

$$\phi_n(c_1,\ldots,c_n) = \prod_{i=1}^{\nu} \left[1 + (2^{a_i} - 1)\frac{\tilde{x}_i}{1 + \tilde{x}_i}\right] - \prod_{i=1}^{\nu} \frac{1}{1 + \tilde{x}_i}. \tag{B.13}$$

In comparison to Eq. (B.2) on page 116 the simplified Eq. (B.10) and Eq. (B.13) are much more convenient for modeling purposes.

Summarizing, the simplifications in the used kinetics include the assumptions of interaction sets (Eq. (B.4)), independent binding of such interaction sets ($r_{\mathbf{v}} = 1\ \forall \mathbf{v}$), and specific regulation strengths of interaction sets. Furthermore,

the regulation strength of multiple interaction sets is the product of individual strengths (Eq. (B.7)).

For the example given in Figure 2.2, the simplified transcription function of a gene with three transcription factors $TF_1$, $TF_2$, $TF_3$, where $TF_1$ is an inhibitor and $TF_2$ and $TF_3$ are in one interaction set, reads with Eq. (B.10)

$$\phi_3(c_1, c_2, c_3) = \left[1 + (2^{a_1} - 1)\frac{\tilde{x}_1}{1 + \tilde{x}_1}\right]\left[1 + (2^{a_2} - 1)\frac{\tilde{x}_2\tilde{x}_3}{1 + \tilde{x}_2\tilde{x}_3}\right]. \quad \text{(B.14)}$$

The regulation strengths are $a_1 < 0$ (Gene1) and $a_2 > 0$ (interaction set). Note that the normalized TF concentration $\{\tilde{x}_i\}$ are used.

# APPENDIX C

# Technical Aspects of GeNGe

## C.1 Behind the Web Interface

GeNGe is designed as a Web application implemented in Zope[1]. Zope is a object-oriented Web application server written in the Python programming language. The application GeNGe is written as a Zope-product and it is as well written in Python. It uses PyBioS[2] (Wierling et al., 2007) for model generation via several application programming interface (API) methods. The connection to PyBioS is shown in Figure C.1. Network structures are generated and kinetic schemata are assigned to each network within GeNGe by users (step ① and ②). Given a network structure and a kinetic schema a mathematical model is set up in PyBioS (step ③). For that, instances of mRNAs, proteins, and enzymes are subsequently included in the model. Furthermore, reactions according to the kinetic schema are assigned to each component. Transcriptional regulatory interactions are included which are defined by the network structure. A mathematical model based on ordinary differential equations is generated within PyBioS and can be exported from PyBioS by Web services API. Once a model is generated simulations can be performed. For that, simulation parameters, such as kinetic parameters and initial conditions are selected by the user (step ④). Subsequent, parameters of the mathematical model are set accordingly and the ODE system is numerically solved by the solver LSODA (Hindmarsh, 1983,

[1] `http://www.zope.org`.

[2] `http://pybios.molgen.mpg.de`.

Petzold, 1983) (step ⑤). Results can be visualized and downloaded from the website (step ⑥).

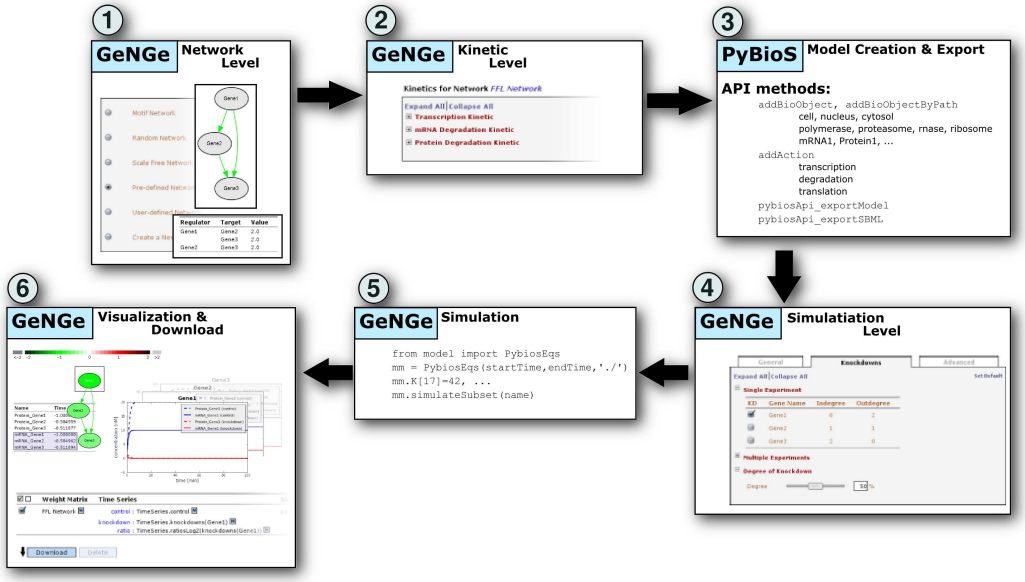



Figure C.1 | **Connection to PyBioS.** GeNGe uses several API methods provided by PyBioS for model generation. An ODE system is generated within PyBioS according to selected network structures and kinetic schemata.

## C.2 Scaling Behavior

For the determination of the scaling behavior of the network generation, i.e. construction of the network topology, I generated ten networks of each of the network types motif network, random network, and scale-free network with different sizes, i.e., number of nodes, varying from 100 to 500 and those values are averaged. Each node corresponds to a single mRNA and its respective protein. For the network generation default parameters are used. The scaling behavior (see Figure C.2) shows a slightly quadratic behavior.

For the determination of the time scaling of the data generation, i.e. generating the mathematical model and solving the ODE system, I used from the above generated networks two networks of each of the network types and performed six time course simulations, respectively. I averaged over all simulations for networks with the same size. Once a mathematical model is generated, the ODE system is used for subsequent simulations. Therefore, I distinguish between model generation, i.e. set up of the ODE system and data generation, i.e. solving the mathematical system. The mathematical system contains

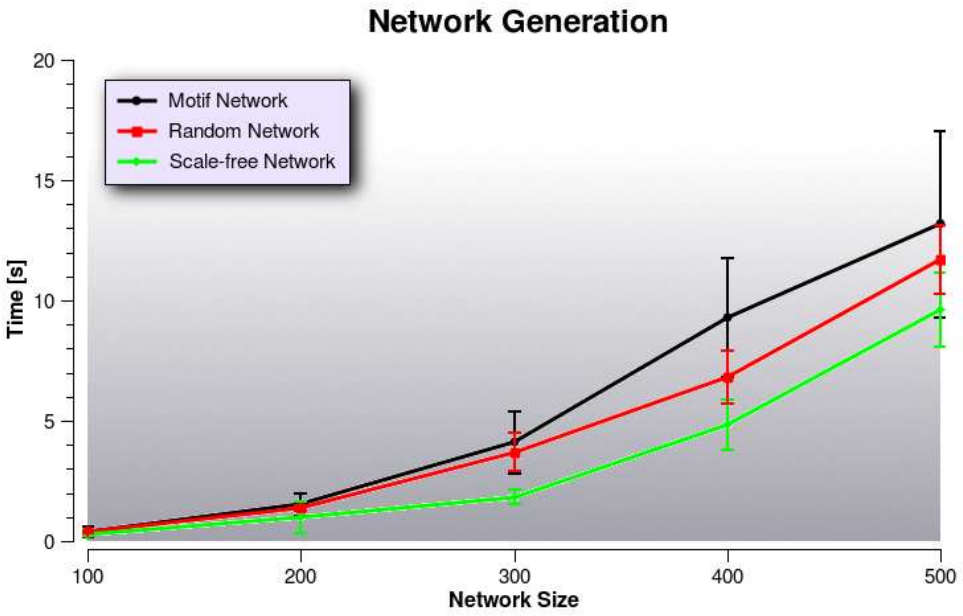



Figure C.2 | **Scaling behavior of GeNGe for network generation.** Network generation is the construction of network topology.

twice as much equations as nodes, since for each node an mRNA and a protein exists. For the simulations, I used also the default parameters settings. The time scaling (see Figure C.3) shows for the model and data generation a quadratic behavior, respectively.

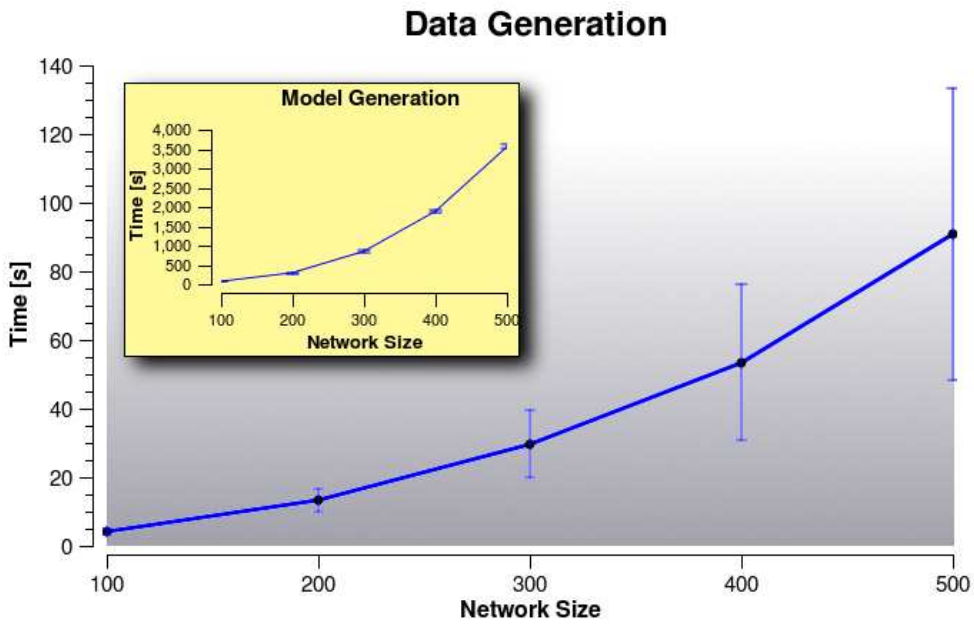



Figure C.3 | **Scaling behavior of GeNGe for Data Generation.** Model generation is the set up of the ODE system, based on the network topology. During data generation the ODE is solved numerically for a specific parameter set.

# APPENDIX D

# BPTT Algorithm

## D.1 Additive Model

In the following the detailed derivation of the BPTT algorithm proposed by Werbos (1990) is shown and applied to the model of gene regulation with neural networks. To recall, the model of gene regulation with $N$ genes is given by

$$y_i[t+\Delta t] = y_i[t] + \Delta t\,[a_i S(z_i[t+\Delta t]) - d_i y_i[t]] \quad \forall i \in \mathcal{N} \tag{D.1}$$

$$\text{with } z_i[t+\Delta t] = \sum_{j\in\mathcal{N}} w_{ij} y_j[t] + b_i. \tag{D.2}$$

The regulatory effect of all connected nodes are weighted and summed up (additive model). This defines the input strength and is passed through an activation function given by

$$S(x) = \frac{1}{1+e^{-x}} \tag{D.3}$$

and its derivation with respect to x

$$\frac{\partial S(x)}{\partial x} = S'(x) = S(x)[1-S(x)]. \tag{D.4}$$

The deviation of newly calculated values, $\mathbf{y}$, with the model and a certain parameter set from the given data, $\hat{\mathbf{y}}$, is given by the error function

$$E(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{2} \sum_t \sum_i \left( y_i[t] - \hat{y}_i[t] \right)^2 . \tag{D.5}$$

In every iteration step of the learning process, each parameter $q \in \{\mathbf{W}, \mathbf{a}, \mathbf{b}, \mathbf{d}\}$ is modified according to

$$q \longrightarrow q + \Delta q. \tag{D.6}$$

The parameter modification is given by

$$\Delta q = -\eta_q \frac{\partial^+ E}{\partial q}, \tag{D.7}$$

with a parameter specific learning rate, $\eta_q$. The operator $\partial^+ / \partial q$ defines a special derivation which can only be applied on systems of ordered variables, $\{Q_1, Q_2, \ldots, Q_k, \ldots, Q_l, \ldots, Q_m, \ldots\}$, with the property that a variable $Q_l$ may be dependent on one or more variables $Q_k$ with $k < l$ and is not dependent on any variable $Q_m$ with $m > l$. The ordered derivation is defined as

$$\frac{\partial^+}{\partial Q_k} = \frac{\partial}{\partial Q_k} + \sum_{l>k} \frac{\partial Q_l}{\partial Q_k} \frac{\partial^+}{\partial Q_l}. \tag{D.8}$$

The ordered derivation considers not only the usual partial derivation with respect to a certain variable (first term), but also the derivations of all variables which are dependent on this variable. That means, that a derivation of a function with respect to a variable which has strong influences on other variables (indicated by large partial deviations) may be distinctly larger than the derivation of this variable without any influences on other variables.

The BPTT algorithm uses this concept of ordered derivation to obtain the parameter modifications. The system given in Eq. (D.1) induces ordered variables with respect to the time such that $y_i[t + \Delta t]$ is dependent on the variables $\{t, y_1[t], \ldots, y_N[t], w_{i1}, \ldots, w_{iN}, a_i, b_i, d_i\}$. The error function given in Eq. (D.5) can be considered to be dependent on all the variables $\{y_1[t = 0], \ldots, y_N[t = T]\}$. The ordered derivation in Eq. (D.7) of the error function with respect to a network parameter $q$ is then

$$\frac{\partial^+ E}{\partial q} = \sum_t \frac{\partial y_i[t + \Delta t]}{\partial q} \frac{\partial^+ E}{\partial y_i[t + \Delta t]}. \tag{D.9}$$

The error function does not depend explicitly on any parameter $q$, therefore, the usual derivation of $E$ with respect to $q$ is zero.

The ordered derivations of the error function with respect to the individual parameters are determined from Eq. (D.9) and Eq. (D.1) as follows, starting with an arbitrary weight $w_{ij}$

$$\begin{aligned}
\frac{\partial^+ E}{\partial w_{ij}} &= \sum_t \frac{\partial y_i[t+\Delta t]}{\partial w_{ij}} \frac{\partial^+ E}{\partial y_i[t+\Delta t]}. \\
&= \sum_t \Delta t\, a_i \frac{\partial S(z_i[t+\Delta t])}{\partial w_{ij}} \frac{\partial^+ E}{\partial y_i[t+\Delta t]} \\
&= \sum_t \Delta t\, a_i \frac{\partial z_i[t+\Delta t]}{\partial w_{ij}} \frac{\partial S(z_i[t+\Delta t])}{\partial z_i[t+\Delta t]} \frac{\partial^+ E}{\partial y_i[t+\Delta t]} \\
&= \sum_t \Delta t\, a_i y_j[t] S'([z_i[t+\Delta t]) \frac{\partial^+ E}{\partial y_i[t+\Delta t]} \\
&= \sum_t y_j[t] \Delta_i[t+\Delta t] \qquad \text{(D.10)}
\end{aligned}$$

with the substitutions

$$\Delta_i[t+\Delta t] = \Delta t\, a_i S'(z_i[t+\Delta t]) \delta_i[t+\Delta t] \qquad \text{(D.11)}$$

$$\delta_i[t+\Delta t] = \frac{\partial^+ E}{\partial y_i[t+\Delta t]}. \qquad \text{(D.12)}$$

The usual derivation chain rule is several times applied. For the other parameters, the derivation is similar and leads to

$$\frac{\partial^+ E}{\partial a_i} = \sum_t \Delta t\, S(z[t]) \delta_i[t+\Delta t] \qquad \text{(D.13)}$$

$$\frac{\partial^+ E}{\partial b_j} = \sum_t \Delta_i[t+\Delta t] \qquad \text{(D.14)}$$

$$\frac{\partial^+ E}{\partial d_i} = -\sum_t \Delta t\, y_j[t] \delta_i[t+\Delta t]. \qquad \text{(D.15)}$$

An arbitrary $y_j[t+\Delta t]$ is dependent of all $\{y_0[t], \ldots, y_N[t]\}$, hence, any $y_j[t+$

$\Delta t]$ is dependent of an arbitrary $y_i[t]$. From that the substitution $\delta_i[t]$ arise from

$$\begin{aligned}
\delta_i[t] &= \frac{\partial^+ E}{\partial y_i[t]} \\
&= \frac{\partial E}{\partial y_i[t]} + \sum_j \frac{\partial y_j[t+\Delta t]}{\partial y_i[t]} \frac{\partial^+ E}{\partial y_j[t+\Delta t]} \\
&= \frac{\partial E}{\partial y_i[t]} + \sum_j \frac{\partial y_j[t+\Delta t]}{\partial y_i[t]} \, \delta_j[t+\Delta t] \\
&= y_i[t] - \hat{y}_i[t] + \\
&\quad + (1 - \Delta t \, d_i)\delta_i[t+\Delta t] + \sum_j \Delta t \, a_j \frac{\partial S(z_j[t+\Delta t])}{\partial y_i[t]} \, \delta_j[t+\Delta t] \\
&= y_i[t] - \hat{y}_i[t] + \\
&\quad + (1 - \Delta t \, d_i)\delta_i[t+\Delta t] + \\
&\quad + \sum_j \Delta t \, a_j \frac{\partial z_j[t+\Delta t]}{\partial y_i[t]} \frac{\partial S(z_j[t+\Delta t])}{\partial z_j[t+\Delta t]} \, \delta_j[t+\Delta t] \\
&= y_i[t] - \hat{y}_i[t] + (1 - \Delta t \, d_i)\delta_i[t+\Delta t] + \sum_j w_{ij} \Delta_j[t+\Delta t] \qquad \text{(D.16)}
\end{aligned}$$

The following parameter modifications have been derived for the BPTT algorithm applied on neural networks

$$\Delta w_{ij} = -\sum_t \eta_{ij}^{(w)} y_j[t] \Delta_i[t+\Delta t] \qquad \text{(D.17a)}$$

$$\Delta a_i = -\sum_t \Delta t \, \eta_i^{(a)} S_i(z_i[t+\Delta t]) \delta_i[t+\Delta t] \qquad \text{(D.17b)}$$

$$\Delta b_i = -\sum_t \eta_i^{(b)} \Delta_i[t+\Delta t] \qquad \text{(D.17c)}$$

$$\Delta d_i = \sum_t \Delta t \, \eta_i^{(d)} y_i[t] \delta_i[t+\Delta t] \qquad \text{(D.17d)}$$

with

$$\Delta_i[t+\Delta t] = \Delta t \, a_i S_i'(z_i[t+\Delta t]) \delta_i[t+\Delta t] \qquad \text{(D.18a)}$$

$$\delta_i[t] = y_i[t] - \hat{y}_i[t] + (1 - \Delta t \, d_i)\delta_i[t+\Delta t] + \sum_j w_{ji} \Delta_j[t+\Delta t] \qquad \text{(D.18b)}$$

and the boundary condition $\delta_i[t] = 0$ for $\forall t > t_T$ and $\forall i$.

## D.2 Multiplicative Model

In the following a new activation function is assumed

$$\tilde{S}_i(y_1[t],\ldots,y_N[t]) = \prod_{j\in\mathcal{N}} \left[1 + (2^{w_{ij}} - 1)\frac{y_j[t]}{1+y_j[t]}\right]. \tag{D.19}$$

The formalism as described above can be applied on the new activation function, Eq. (D.19) to obtain new parameter modification terms. With Eq. (D.1) it follows for the weights terms

$$\frac{\partial^+ E}{\partial w_{ij}} = \sum_t \frac{\partial y_i[t+\Delta t]}{\partial w_{ij}} \frac{\partial^+ E}{\partial y_i[t+\Delta t]} \tag{D.20}$$

$$= \sum_t \Delta t \cdot a_i \frac{\partial \tilde{S}_i(\mathbf{y})}{\partial w_{ij}} \delta_i[t+\Delta t] \tag{D.21}$$

with $\tilde{S}_i(\mathbf{y} = \tilde{S}_i(y_1[t],\ldots,y_N[t])$ and $\delta_i[t+\Delta t] = \partial^+ E/\partial y_i[t+\Delta t]$. The derivations of the new activation functions with respect to an arbitrary weight, $w_{ij}$ is

$$\frac{\partial \tilde{S}(\mathbf{y})}{\partial w_{ij}} = \frac{\tilde{S}(\mathbf{y})}{1 + (2^{w_{ij}} - 1)\frac{y_j[t]}{1+y_j[t]}} \frac{\ln 2 \cdot 2^{w_{ij}}}{1+y_j[t]} \tag{D.22}$$

$$= \ln 2 \cdot \tilde{S}(\mathbf{y}) \frac{2^{w_{ij}} y_j[t]}{1 + 2^{w_{ij}} y_j[t]}. \tag{D.23}$$

This inserted to Eq. (D.21) yields to

$$\frac{\partial^+ E}{\partial w_{ij}} = \sum_t \Delta t \cdot a_i \tilde{S}_i(\mathbf{y}) \delta_i[t+\Delta t] \frac{\ln 2 \cdot 2^{w_{ij}} y_j[t]}{1 + 2^{w_{ij}} y_j[t]} \tag{D.24}$$

$$= \sum_t \Delta_i[t+\Delta t] \frac{\ln 2 \cdot 2^{w_{ij}} y_j[t]}{1 + 2^{w_{ij}} y_j[t]} \tag{D.25}$$

with

$$\Delta_i[t+\Delta t] = \Delta t \cdot a_i \tilde{S}_i(\mathbf{y}) \delta_i[t+\Delta t]. \tag{D.26}$$

The error, $\delta_i[t]$, is determined by

$$\delta_i[t] = \frac{\partial^+ E^+}{\partial y_i[t]} \tag{D.27}$$

$$= \frac{\partial^+ E}{\partial y_i[t]} + \sum_j \frac{\partial y_j[t+\Delta t]}{\partial y_i[t]} \delta_j[t+\Delta t] \tag{D.28}$$

$$= y_i[t] - \hat{y}_i[t] + (1 - \Delta t d_i)\delta_i[t+\Delta t] + \sum_j \Delta t\, a_j \frac{\partial S_j(\mathbf{y})}{\partial y_i[t]} \delta_j[t+\Delta t] \tag{D.29}$$

With

$$\frac{\partial \tilde{S}(\mathbf{y})}{\partial y_j[t]} = \frac{\tilde{S}(\mathbf{y})}{1 + (2^{w_{ij}} - 1)\frac{y_j[t]}{1+y_j[t]}} \frac{2^{w_{ij}} - 1}{(1 + y_j[t])^2} \tag{D.30}$$

$$= \tilde{S}(\mathbf{y}) \frac{2^{w_{ij}} - 1}{1 + y_j[t] + 2^{w_{ij}} y_j[t] + +2^{w_{ij}} y_j[t]^2} \tag{D.31}$$

follows

$$\delta_i[t] = y_i[t] - \hat{y}_i[t] + (1 - \Delta t d_i)\delta_i[t+\Delta t] + + \sum_j \Delta t\, a_j \tilde{S}(\mathbf{y}) \frac{2^{w_{ij}} - 1}{1 + y_j[t] + 2^{w_{ij}} y_j[t] + +2^{w_{ij}} y_j[t]^2} \delta_j[t+\Delta t] \tag{D.32}$$

$$= y_i[t] - \hat{y}_i[t] + (1 - \Delta t d_i)\delta_i[t+\Delta t] + + \sum_j \Delta_j[t+\Delta t] \frac{2^{w_{ij}} - 1}{1 + y_j[t] + 2^{w_{ij}} y_j[t] + +2^{w_{ij}} y_j[t]^2}. \tag{D.33}$$

The modification terms for the other parameters, **a**, **b**, and **d**, are similar to Eq. (D.17) with the new error function Eq. (D.33).

# APPENDIX E

# Classification Matrix

Matrices which represent network topologies can be compared by means of a classification table. It is assumed, that the matrices are ternary, i.e., $w_{ij} \in \{-1, 0, 1\}\ \forall i, j \in \mathcal{N}$, representing activation, inhibition, or non-regulation. One matrix serves as a true model and the other one is compared with that. Respective matrix entries are compared individually and classified according to the classification matrix shown in Figure E.1. As an extension to a binary classification a nomenclature is introduced to describe different match possibilities: true regulation (TR) for correctly classified regulation, true zero (TZ) for correctly identified non-regulation, false regulation (FR) for incorrectly identified regulation, false zero (FZ) for interaction which could not be revealed, and false interaction (FI) for identified interaction with false type (activation or inhibition).

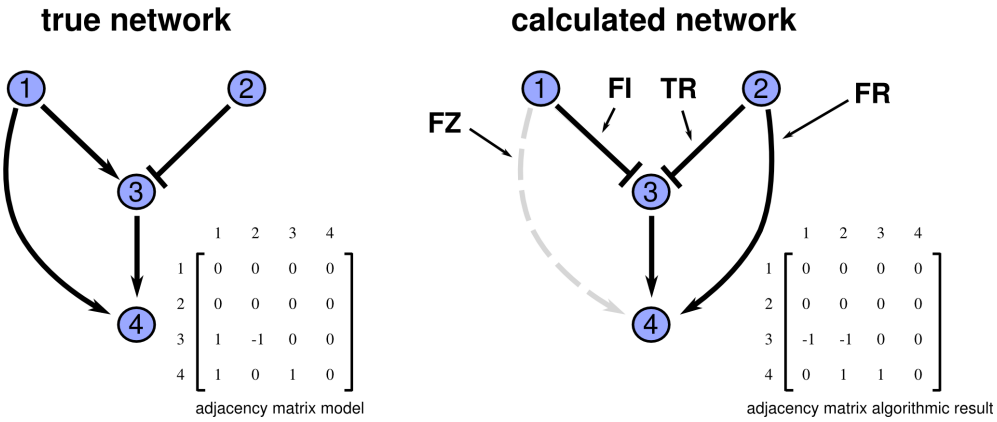
Several classification measures can be calculated, such as sensitivity, specificity, precision, and other.

$$\text{sen} := \frac{\text{TR}}{\text{TR} + \text{FZ} + \text{FI}} \tag{E.1}$$

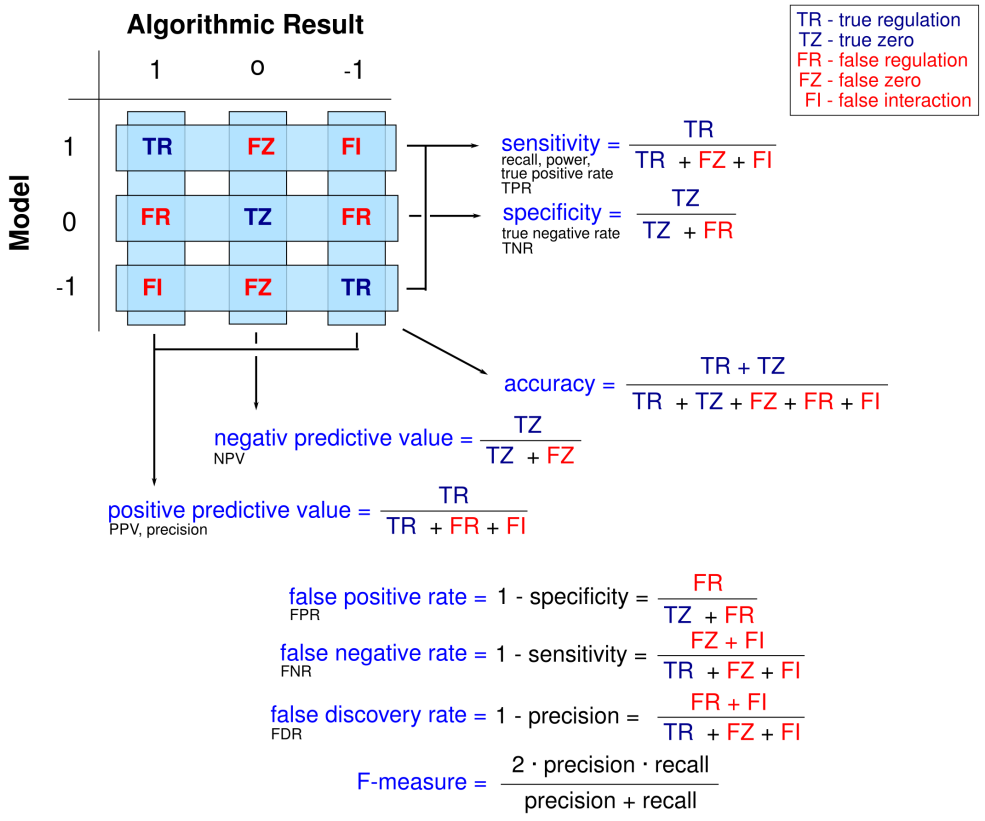
$$\text{spe} := \frac{\text{TZ}}{\text{TZ} + \text{FR}} \tag{E.2}$$

$$\text{pre} := \frac{\text{TR}}{\text{TR} + \text{FR} + \text{FI}} \tag{E.3}$$

These measures are used to evaluate the similarity between two matrices representing a topology. Especially for comparative studies of reverse engineering or similar methods these measures are essential for assessing performances.




(a) True vs. calculated network.




(b) Classification matrix and definitions

Figure E.1 | **Classification matrix and definitions. a** | Example of a true (model) network and a calculated network. The adjacency matrix represents the network structure. **b** | The gene regulatory models have three discrete states (1: activation, $-1$: inhibition, 0: non-regulation). The kind of regulation (activation or inhibition) in the classification is considered in the classification of the results according to the model: TR, true regulation, TZ, true zero, FR, false regulation, FZ, false zero, and FI, false interaction.

# APPENDIX F

# Publications


Hache, H., Wierling, C., Lehrach, H., and Herwig, R. (2009) GeNGe: Systematic generation of gene regulatory networks. *Bioinformatics*, **25(9)**:1205–1207.

Hache, H., Lehrach, H., and Herwig, R. (2009) Reverse engineering of gene regulatory networks: A comparative study. *EURASIP J Bioinform Syst Biol*. **2009**

Hache, H. (2008) *Handbook of Research on Systems Biology Applications in Medicine*, volume II, Chapter XXIX: Methods for Reverse Engineering of Gene Regulatory Networks. pp 497–515

Hache, H., Wierling, C., Lehrach, H., and Herwig, R. (2007) Reconstruction and validation of gene regulatory networks with neural networks. *Proceedings of the 2nd Foundations of Systems Biology in Engineering Conference (FOSBE)*, Stuttgart, pp. 319–324.


Berlin, 22. Juni 2009

Hendrik Hache

# Abbreviations & Symbols

## List of Abbreviations

| Abbreviation | Description |
|---|---|
| AUC | area under curve |
| bp | base pair |
| BPTT | Backpropagation Through Time Algorithm |
| CV | coefficient of variation |
| DBN | dynamic Bayesian network method |
| DE | differential euqation |
| DNA | deoxyribonucleic acid |
| GGM | graphical Gaussian model method |
| GRN | gene regulatory model |
| kp | kilo base pair |
| miRNA | micro RNA |
| mRNA | messenger RNA |
| MI | mutual information method |
| NN | neural network method (refers to GNRevealer) |
| ODE | ordinary differential equation |
| PC | Pearson (partial) correlation method |
| RNA | ribonucleic acid |
| RNAi | RNA interference |
| ROC | receiver operation characteristic |
| SC | Spearman (partial) correlation method |
| SSM | state space model method |
| TF | transcription factor |

# List of Symbols

Mathematical symbols are listed along with the page where it is defined.

| Symbol | | Description |
|---|---|---|
| $\{a, b, c\}$ | | set of objects, such as nodes or edges |
| $[A]$ | | concentration of an object |
| $\dim(\mathcal{A})$ | | dimension of set $\mathcal{A}$, i.e., number of elements |
| $\mathbf{a}$ | * | parameter vector in neural network model (p. 58) |
| $\mathbf{A}$ | * | vector of regulation strengths associated to transcription factors (p. 29) |
| $\mathbf{b}$ | * | parameter vector in neural network model (p. 58) |
| $\mathbf{c}$ | * | vector of concentration values of transcription factors (p. 26) |
| $d$ | | distance measure between a network and a source network (p. 67) |
| $\mathbf{d}$ | | parameter vector in neural network model (p. 58) |
| $\mathcal{D}$ | | set of data, such as expression profiles for each gene in a GRN (p. 77) |
| $\mathcal{E}$ | | set of edges in a GRN (p. 11) |
| $\eta_q$ | | learning rate associated to parameter $q$ (p. 63) |
| $\mathcal{F}$ | | set of functions in a graphical structure (p. 17) |
| $\mathcal{G}$ | | graphical structure (p. 11) |
| $\mathcal{N}$ | | set of indices of nodes in a GRN (p. 11) |
| $\mathcal{N}_{R_i}$ | | set of indices of regulators of gene $i$ in a GRN (p. 12) |
| $\mathcal{N}_{T_i}$ | | set of indices of targets of gene $i$ in a GRN (p. 12) |
| $k^i_{\text{in}}$ | | number of in-links of node $i$ (p. 13) |
| $k^i_{\text{out}}$ | | number of out-links of node $i$ (p. 13) |
| $\mathbf{K}$ | * | set of normalization factors for transcription factor concentrations (p. 29) |
| $K_M$ | * | Michaelis-Menten constant (p. 25) |
| $N$ | | number of nodes in a GRN (p. 11) |
| ppr | | precision (p. 66) |
| $\phi_n$ | * | transcription function (p. 25) |
| $\mathcal{Q}$ | | set of parameters in a gene regulatory model (p. 17) |
| $\mathcal{R}_i$ | | set of regulators of node $i$ (p. 12) |
| $\mathbf{r}_i$ | | value vector associated to $\mathcal{R}_i$ (p. 18) |
| $S_i$ | | activation function in neural network model (p. 58) |
| sen | | sensitivity (p. 66) |
| spe | | specificity (p. 66) |

*Continued on next page*

| Symbol | Description |
|---|---|
| $\mathcal{T}_i$ | set of indices of transcription factors of the gene in context (p. 12) |
| $\mathcal{V}$ | Set of nodes or vertices in a GRN (p. 11) |
| $\mathbf{W}$ | matrix of weigh values, such that $w_{ij} \in \mathbf{W}$ represents the value of edge from node $j$ to $i$ (p. 11) |
| $\mathcal{W}$ | list of regulations with regulation strengths (p. 11) |
| $\mathbf{x}$ | value vector of variables $\mathcal{X}$ (p. 17) |
| $\mathcal{X}$ | set of variables associated to nodes in a GRN (p. 17) |
| $\mathcal{X}_{\mathcal{R}_i}$ | set of variables associated to nodes in $\mathcal{R}_i$ (p. 17) |

* Symbol may have an additional index for indicating the corresponding node in network.

# Bibliography

T. Aho, O.-P. Smolander, J. Niemi, and O. Yli-Harja. Rmbntoolbox: random models for biochemical networks. *BMC Syst Biol*, 1:22, 2007. doi: 10.1186/1752-0509-1-22.

T. Aittokallio and B. Schwikowski. Graph-based methods for analysing networks in cell biology. *Brief Bioinform*, 7(3):243–255, Sep 2006. doi: 10.1093/bib/bbl022.

H. Akima. A New Method of Interpolation and Smooth Curve Fitting Based on Local Procedures. *J. ACM*, 17(4):589 – 602, October 1970.

R. Albert. Scale-free networks in cell biology. *J Cell Sci*, 118(Pt 21):4947–4957, Nov 2005. doi: 10.1242/jcs.02714.

R. Albert and A.-L. Barabási. Statistical mechanics of complex networks. *Rev. Mod. Phys.*, 74(1):47–97, Jan 2002. doi: 10.1103/RevModPhys.74.47.

B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter. *Molecular Biology of the Cell*. Garland Science, Taylor & Francis Group, LLC, an informa business, 270 Madison Avenue, New York NY 10016, USA, and 2 Park Square, Milton Park, Abingdon, OX14 4RN, UK, 5th edition, 2008.

U. Alon. Network motifs: theory and experimental approaches. *Nat Rev Genet*, 8(6):450–461, Jun 2007. doi: 10.1038/nrg2102.

Y. Artzy-Randrup, S. J. Fleishman, N. Ben-Tal, and L. Stone. Comment on "network motifs: simple building blocks of complex networks" and "superfamilies of evolved and designed networks". *Science*, 305(5687):1107; author reply 1107, Aug 2004. doi: 10.1126/science.1099334.

K. Baba, R. Shibata, and M. Sibuya. Partial correlation and conditional correlation as measures of conditional independence. *Australian & New Zealand Journal of Statistics*, 46(4):657–664, 2004. ISSN 1369-1473. doi: 10.1111/j.1467-842X.2004.00360.x.

Y. Babaie, R. Herwig, B. Greber, T. C. Brink, W. Wruck, D. Groth, H. Lehrach, T. Burdon, and J. Adjaye. Analysis of oct4-dependent transcriptional networks regulating self-renewal and pluripotency in human embryonic stem cells. *Stem Cells*, 25(2):500–510, Feb 2007. doi: 10.1634/stemcells.2006-0426.

M. Bansal, G. D. Gatta, and D. di Bernardo. Inference of gene regulatory networks and compound mode of action from time course gene expression profiles. *Bioinformatics*, 22(7):815–822, Apr 2006. doi: 10.1093/bioinformatics/btl003.

M. Bansal, V. Belcastro, A. Ambesi-Impiombato, and D. di Bernardo. How to infer gene networks from expression profiles. *Mol Syst Biol*, 3:78, 2007. doi: 10.1038/msb4100120.

Barabási and Albert. Emergence of scaling in random networks. *Science*, 286 (5439):509–512, Oct 1999.

A.-L. Barabási and Z. N. Oltvai. Network biology: understanding the cell's functional organization. *Nat Rev Genet*, 5(2):101–113, Feb 2004. doi: 10.1038/nrg1272.

K. Basso, A. A. Margolin, G. Stolovitzky, U. Klein, R. Dalla-Favera, and A. Califanon. Reverse engineering of regulatory networks in human B cells. *Nat Genet*, 37(4):382–390, Apr 2005. doi: 10.1038/ng1532.

V. Batagelj and U. Brandes. Efficient generation of large random networks. *Phys. Rev. E*, 71(3):036113, Mar 2005. doi: 10.1103/PhysRevE.71.036113.

M. J. Beal, F. Falciani, Z. Ghahramani, C. Rangel, and D. L. Wild. A bayesian approach to reconstructing genetic regulatory networks with hidden factors. *Bioinformatics*, 21(3):349–356, Feb 2005. doi: 10.1093/bioinformatics/bti014.

J. Berg and M. Lässig. Local graph alignment and motif search in biological networks. *Proc Natl Acad Sci U S A*, 101(41):14689–14694, Oct 2004. doi: 10.1073/pnas.0305199101.

K. Berns, E. M. Hijmans, J. Mullenders, T. R. Brummelkamp, A. Velds, M. Heimerikx, R. M. Kerkhoven, M. Madiredjo, W. Nijkamp, B. Weigelt, R. Agami, W. Ge, G. Cavet, P. S. Linsley, R. L. Beijersbergen, and R. Bernards. A large-scale rnai screen in human cells identifies new components of the p53 pathway. *Nature*, 428(6981):431–437, Mar 2004. doi: 10.1038/nature02371.

B. Bollobás, C. Borgs, J. Chayes, and O. Riordan. Directed scale-free graphs. In *SODA '03: Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 132–139, Philadelphia, PA, USA, 2003. Society for Industrial and Applied Mathematics. ISBN 0-89871-538-5.

L. A. Boyer, T. I. Lee, M. F. Cole, S. E. Johnstone, S. S. Levine, J. P. Zucker, M. G. Guenther, R. M. Kumar, H. L. Murray, R. G. Jenner, D. K. Gifford, D. A. Melton, R. Jaenisch, and R. A. Young. Core transcriptional regulatory circuitry in human embryonic stem cells. *Cell*, 122(6):947–956, Sep 2005. doi: 10.1016/j.cell.2005.08.020.

J. C. Bryne, E. Valen, M.-H. E. Tang, T. Marstrand, O. Winther, I. da Piedade, A. Krogh, B. Lenhard, and A. Sandelin. Jaspar, the open access database of transcription factor-binding profiles: new content and tools in the 2008 update. *Nucleic Acids Res*, 36(Database issue):D102–D106, Jan 2008. doi: 10.1093/nar/gkm955.

N. E. Buchler, U. Gerland, and T. Hwa. Nonlinear protein degradation and the function of genetic circuits. *Proc Natl Acad Sci U S A*, 102(27):9559–9564, Jul 2005. doi: 10.1073/pnas.0409553102.

L. Cai, N. Friedman, and S. X. Xie. Stochastic protein expression in individual cells at the single molecule level. *Nature*, 440(7082):358–362, 2006. ISSN 0028-0836. doi: 10.1038/nature04599.

D. Camacho, P. V. Licona, P. Mendes, and R. Laubenbacher. Comparison of reverse-engineering methods using an in silico network. *Ann N Y Acad Sci*, 1115:73–89, Dec 2007. doi: 10.1196/annals.1407.006.

K. Chen and N. Rajewsky. The evolution of gene regulation by transcription factors and microrna s. *Nat Rev Genet*, 8(2):93–103, Feb 2007. doi: 10.1038/nrg1990.

T. Chen, H. L. He, and G. M. Church. Modeling gene expression with differential equations. *Pac Symp Biocomput*, pages 29–40, 1999.

T. Chen, V. Filklov, and S. S. Skiena. Identifying gene regulatory networks from experimental data. *Parallel Comput.*, 27(1-2):141–162, 2001. ISSN 0167-8191. doi: 10.1016/S0167-8191(00)00092-2.

J. L. Cherry and F. R. Adler. How to make a biological switch. *J Theor Biol*, 203 (2):117–133, Mar 2000. doi: 10.1006/jtbi.2000.1068.

D. M. Chickering. Learning Equivalence Classes Of Bayesian-Network Structures. *J. Mach. Learn. Res.*, 2:445–498, 2002.

R. J. Cho, M. J. Campbell, E. A. Winzeler, L. Steinmetz, A. Conway, L. Wodicka, T. G. Wolfsberg, A. E. Gabrielian, D. Landsman, D. J. Lockhart, and R. W. Davis. A genome-wide transcriptional analysis of the mitotic cell cycle. *Mol Cell*, 2(1):65–73, Jul 1998.

F. Chung and L. Lu. The average distances in random graphs with given expected degrees. *Proc Natl Acad Sci U S A*, 99(25):15879–15882, Dec 2002. doi: 10.1073/pnas.252631999.

G. F. Cooper and E. Herskovits. A bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 09(4):309–347, October 1992. ISSN 0885-6125.

E. H. Davidson, J. P. Rast, P. Oliveri, A. Ransick, C. Calestani, C.-H. Yuh, T. Minokawa, G. Amore, V. Hinman, C. Arenas-Mena, O. Otim, C. T. Brown, C. B. Livi, P. Y. Lee, R. Revilla, A. G. Rust, Z. jun Pan, M. J. Schilstra, P. J. C. Clarke, M. I. Arnone, L. Rowen, R. A. Cameron, D. R. McClay, L. Hood, and H. Bolouri. A genomic regulatory network for development. *Science*, 295(5560):1669–1678, Mar 2002. doi: 10.1126/science.1069883.

H. de Jong. Modeling and simulation of genetic regulatory systems: a literature review. *J Comput Biol*, 9(1):67–103, 2002. doi: 10.1089/10665270252833208.

A. de la Fuente, N. Bing, I. Hoeschele, and P. Mendes. Discovery of meaningful associations in genomic data using partial correlation coefficients. *Bioinformatics*, 20(18):3565–3574, Dec 2004. doi: 10.1093/bioinformatics/bth445.

S. B.-T. de Leon and E. H. Davidson. Modeling the dynamics of transcriptional gene regulatory networks for animal development. *Dev Biol*, 325(2):317–328, Jan 2009. doi: 10.1016/j.ydbio.2008.10.043.

T. V. den Bulcke, K. V. Leemput, B. Naudts, P. van Remortel, H. Ma, A. Verschoren, B. D. Moor, and K. Marchal. Syntren: a generator of synthetic gene expression data for design and analysis of structure learning algorithms. *BMC Bioinformatics*, 7:43, 2006. doi: 10.1186/1471-2105-7-43.

J. DeRisi, L. Penland, P. O. Brown, M. L. Bittner, P. S. Meltzer, M. Ray, Y. Chen, Y. A. Su, and J. M. Trent. Use of a cdna microarray to analyse gene expression patterns in human cancer. *Nat Genet*, 14(4):457–460, Dec 1996. doi: 10.1038/ng1296-457.

J. L. DeRisi, V. R. Iyer, and P. O. Brown. Exploring the metabolic and genetic control of gene expression on a genomic scale. *Science*, 278(5338):680–686, Oct 1997.

P. D'haeseleer, X. Wen, S. Fuhrman, and R. Somogyi. Linear modeling of mRNA expression levels during CNS development and injury. *Pac Symp Biocomput*, pages 41–52, 1999.

R. Dobrin, Q. K. Beg, A.-L. Barabási, and Z. N. Oltvai. Aggregation of topological motifs in the escherichia coli transcriptional regulatory network. *BMC Bioinformatics*, 5:10, Jan 2004. doi: 10.1186/1471-2105-5-10.

D. Edwards. *Introduction to Graphical Modelling*. Springer, June 2000. ISBN 0387950540.

M. B. Elowitz and S. Leibler. A synthetic oscillatory network of transcriptional regulators. *Nature*, 403(6767):335–338, Jan 2000. doi: 10.1038/35002125.

P. Erdős and A. Rényi. On random graphs. i. *Publicationes Mathematicae*, 6: 290–297, 1959.

D. E. Featherstone and K. Broadie. Wrestling with pleiotropy: genomic and topological analysis of the yeast gene expression network. *Bioessays*, 24(3): 267–274, Mar 2002. doi: 10.1002/bies.10054.

S. Fields and O. Song. A novel genetic system to detect protein-protein interactions. *Nature*, 340(6230):245–246, Jul 1989. doi: 10.1038/340245a0.

A. Fire, S. Xu, M. K. Montgomery, S. A. Kostas, S. E. Driver, and C. C. Mello. Potent and specific genetic interference by double-stranded rna in caenorhabditis elegans. *Nature*, 391(6669):806–811, Feb 1998. doi: 10.1038/35888.

N. Friedman, K. Murphy, and S. Russell. Learning the Structure of Dynamic Probabilistic Networks. In G. Cooper and S. Moral, editors, *UAI '98: Proceedings of the Fourteenth Annual Conference on Uncertainty in Artificial Intelligence*, pages 139–147, Madison, Wisconsin, USA, 1998. Morgan Kaufmann.

N. Friedman, M. Linial, I. Nachman, and D. Pe'er. Using Bayesian Networks to Analyze Expression Data. *J Comput Biol*, 7(3-4):601–620, 2000. doi: 10.1089/106652700750050961.

A. Funahashi, M. Morohashi, H. Kitano, and N. Tanimura. Celldesigner: a process diagram editor for gene-regulatory and biochemical networks. *BIOSILICO*, 1(5):159 – 162, 2003. ISSN 1478-5382. doi: 10.1016/S1478-5382(03)02370-9.

M. Y. Galperin and G. R. Cochrane. Nucleic acids research annual database issue and the nar online molecular biology database collection in 2009. *Nucleic Acids Res*, 37(Database issue):D1–D4, Jan 2009. doi: 10.1093/nar/gkn942.

S. Gama-Castro, V. Jiménez-Jacinto, M. Peralta-Gil, A. Santos-Zavaleta, M. I. Peñaloza-Spinola, B. Contreras-Moreira, J. Segura-Salazar, L. Muñiz-Rascado, I. Martínez-Flores, H. Salgado, C. Bonavides-Martínez, C. Abreu-Goodger, C. Rodríguez-Penagos, J. Miranda-Ríos, E. Morett, E. Merino, A. M. Huerta, L. Treviño-Quintanilla, and J. Collado-Vides. Regulondb (version 6.0): gene regulation model of escherichia coli k-12 beyond transcription, active (experimental) annotated promoters and textpresso navigation. *Nucleic Acids Res*, 36(Database issue):D120–D124, Jan 2008. doi: 10.1093/nar/gkm994.

T. S. Gardner and J. J. Faith. Reverse-engineering transcription control networks. *Physics of Life Reviews*, 2(1):65–88, Mar. 2005.

T. S. Gardner, C. R. Cantor, and J. J. Collins. Construction of a genetic toggle switch in escherichia coli. *Nature*, 403(6767):339–342, Jan 2000. doi: 10.1038/35002131.

T. S. Gardner, D. di Bernardo, D. Lorenz, and J. J. Collins. Inferring genetic networks and identifying compound mode of action via expression profiling. *Science*, 301(5629):102–105, Jul 2003. doi: 10.1126/science.1081900.

D. Geiger and D. Heckerman. Learning gaussian networks. In *UAI*, pages 235–243, 1994.

D. T. Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *Journal of Computational Physics*, 22(4):403–434, December 1976. doi: 10.1016/0021-9991(76)90041-3.

S. B. Gillispie and M. D. Perlman. Enumerating markov equivalence classes of acyclic digraph models. In *Proceedings of the 17th Conference in Artificial Intelligence*, pages 171–177, 2001.

A. Goldbeter and O. Pourquié. Modeling the segmentation clock as a network of coupled oscillations in the notch, wnt and fgf signaling pathways. *J Theor Biol*, 252(3):574–585, Jun 2008. doi: 10.1016/j.jtbi.2008.01.006.

P. J. Goss and J. Peccoud. Quantitative modeling of stochastic systems in molecular biology by using stochastic petri nets. *Proc Natl Acad Sci U S A*, 95(12):6750–6755, Jun 1998.

C. Grilly, J. Stricker, W. L. Pang, M. R. Bennett, and J. Hasty. A synthetic gene network for tuning protein degradation in saccharomyces cerevisiae. *Mol Syst Biol*, 3:127, 2007. doi: 10.1038/msb4100168.

K. L. Gunderson, S. Kruglyak, M. S. Graige, F. Garcia, B. G. Kermani, C. Zhao, D. Che, T. Dickinson, E. Wickham, J. Bierle, D. Doucet, M. Milewski, R. Yang,

C. Siegmund, J. Haas, L. Zhou, A. Oliphant, J.-B. Fan, S. Barnard, and M. S. Chee. Decoding randomly ordered dna arrays. *Genome Res*, 14(5):870–877, May 2004. doi: 10.1101/gr.2255804.

H. Hache. Methods for reverse engineering of gene regulatory networks. In A. Daskalaki, editor, *Handbook of Research on Systems Biology Applications in Medicine*, volume II, chapter XXIX, pages 497–515. Medical Information Science Reference, October 2008.

H. Hache, C. Wierling, H. Lehrach, and R. Herwig. Reconstruction and validation of gene regulatory networks with neural networks. In *FOSBE '07: Proceedings of the 2nd Foundations of Systems Biology in Engineering Conference*, pages 319–324, Sep 2007.

H. Hache, H. Lehrach, and R. Herwig. Reverse engineering of gene regulatory networks: A comparative study. *EURASIP J Bioinform Syst Biol*, 2009, 2009a. doi: doi:10.1155/2009/617281.

H. Hache, C. Wierling, H. Lehrach, and R. Herwig. Genge: Systematic generation of gene regulatory networks. *Bioinformatics*, 25(9):1205–1207, Feb 2009b. doi: 10.1093/bioinformatics/btp115.

D. A. Hall, J. Ptacek, and M. Snyder. Protein microarray technology. *Mech Ageing Dev*, 128(1):161–167, Jan 2007. doi: 10.1016/j.mad.2006.11.021.

J.-D. J. Han, D. Dupuy, N. Bertin, M. E. Cusick, and M. Vidal. Effect of sampling on topology predictions of protein-protein interaction networks. *Nat Biotechnol*, 23(7):839–844, Jul 2005. doi: 10.1038/nbt1116.

L. H. Hartwell, J. J. Hopfield, S. Leibler, and A. W. Murray. From molecular to modular cell biology. *Nature*, 402(6761 Suppl):C47–C52, Dec 1999. doi: 10.1038/35011540.

J. Hasty, D. McMillen, F. Isaacs, and J. J. Collins. Computational studies of gene regulatory networks: in numero molecular biology. *Nat Rev Genet*, 2(4): 268–279, Apr 2001. doi: 10.1038/35066056.

D. Heckerman, D. Geiger, and D. M. Chickering. Learning bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20(3): 197–243, September 1995. ISSN 0885-6125.

A. D. Hill, J. R. Tomshine, E. M. B. Weeding, V. Sotiropoulos, and Y. N. Kaznessis. Synbioss: the synthetic biology modeling suite. *Bioinformatics*, 24(21): 2551–2553, Nov 2008. doi: 10.1093/bioinformatics/btn468.

A. Hindmarsh. Odepack, a systematized collection of ode solvers. In R. Stepleman and et al., editors, *Scientific computing*, pages 55–64. North-Holland, Amsterdam, 1983.

T. Holen, M. Amarzguioui, M. T. Wiiger, E. Babaie, and H. Prydz. Positional effects of short interfering rnas targeting the human coagulation trigger tissue factor. *Nucleic Acids Res*, 30(8):1757–1766, Apr 2002.

S. Hoops, S. Sahle, R. Gauges, C. Lee, J. Pahle, N. Simus, M. Singhal, L. Xu, P. Mendes, and U. Kummer. Copasi–a complex pathway simulator. *Bioinformatics*, 22(24):3067–3074, Dec 2006. doi: 10.1093/bioinformatics/btl485.

M. Hucka, A. Finney, H. M. Sauro, H. Bolouri, J. C. Doyle, H. Kitano, A. P. Arkin, B. J. Bornstein, D. Bray, A. Cornish-Bowden, A. A. Cuellar, S. Dronov, E. D. Gilles, M. Ginkel, V. Gor, I. I. Goryanin, W. J. Hedley, T. C. Hodgman, J.-H. Hofmeyr, P. J. Hunter, N. S. Juty, J. L. Kasberger, A. Kremling, U. Kummer, N. L. Novère, L. M. Loew, D. Lucio, P. Mendes, E. Minch, E. D. Mjolsness, Y. Nakayama, M. R. Nelson, P. F. Nielsen, T. Sakurada, J. C. Schaff, B. E. Shapiro, T. S. Shimizu, H. D. Spence, J. Stelling, K. Takahashi, M. Tomita, J. Wagner, J. Wang, and S. B. M. L. Forum. The systems biology markup language (sbml): a medium for representation and exchange of biochemical network models. *Bioinformatics*, 19(4):524–531, Mar 2003.

T. R. Hughes, M. J. Marton, A. R. Jones, C. J. Roberts, R. Stoughton, C. D. Armour, H. A. Bennett, E. Coffey, H. Dai, Y. D. He, M. J. Kidd, A. M. King, M. R. Meyer, D. Slade, P. Y. Lum, S. B. Stepaniants, D. D. Shoemaker, D. Gachotte, K. Chakraburtty, J. Simon, M. Bard, and S. H. Friend. Functional discovery via a compendium of expression profiles. *Cell*, 102(1):109–126, Jul 2000.

T. R. Hughes, M. Mao, A. R. Jones, J. Burchard, M. J. Marton, K. W. Shannon, S. M. Lefkowitz, M. Ziman, J. M. Schelter, M. R. Meyer, S. Kobayashi, C. Davis, H. Dai, Y. D. He, S. B. Stephaniants, G. Cavet, W. L. Walker, A. West, E. Coffey, D. D. Shoemaker, R. Stoughton, A. P. Blanchard, S. H. Friend, and P. S. Linsley. Expression profiling using microarrays fabricated by an inkjet oligonucleotide synthesizer. *Nat Biotechnol*, 19(4):342–347, Apr 2001. doi: 10.1038/86730.

S. Imoto, T. Goto, and S. Miyano. Estimation of genetic networks and functional structures between genes by using Bayesian networks and nonparametric regression. *Pac Symp Biocomput*, pages 175–186, 2002.

S. Imoto, T. Higuchi, T. Goto, K. Tashiro, S. Kuhara, and S. Miyano. Combining microarrays and biological knowledge for estimating gene networks via bayesian networks. *Proc IEEE Comput Soc Bioinform Conf*, 2:104–113, 2003.

F. J. Isaacs, J. Hasty, C. R. Cantor, and J. J. Collins. Prediction and measurement of an autoregulatory genetic module. *Proc Natl Acad Sci U S A*, 100(13):7714–7719, Jun 2003. doi: 10.1073/pnas.1332628100.

S. Ishihara, K. Fujimoto, and T. Shibata. Cross talking of network motifs in gene regulation that generates temporal pulses and spatial stripes. *Genes Cells*, 10 (11):1025–1038, Nov 2005. doi: 10.1111/j.1365-2443.2005.00897.x.

M. Ishiura, S. Kutsuna, S. Aoki, H. Iwasaki, C. R. Andersson, A. Tanabe, S. S. Golden, C. H. Johnson, and T. Kondo. Expression of a gene cluster kaiabc as a circadian feedback process in cyanobacteria. *Science*, 281(5382):1519–1523, Sep 1998.

T. K. Jenssen, A. Laegreid, J. Komorowski, and E. Hovig. A literature network of human genes for high-throughput analysis of gene expression. *Nat Genet*, 28(1):21–28, May 2001. doi: 10.1038/88213.

H. Jeong, B. Tombor, R. Albert, Z. N. Oltvai, and A. L. Barabási. The large-scale organization of metabolic networks. *Nature*, 407(6804):651–654, Oct 2000. doi: 10.1038/35036627.

C. Jiang, Z. Xuan, F. Zhao, and M. Q. Zhang. Tred: a transcriptional regulatory element database, new entries and other development. *Nucleic Acids Res*, 35 (Database issue):D137–D140, Jan 2007. doi: 10.1093/nar/gkl1041.

M. Kaern, W. J. Blake, and J. J. Collins. The engineering of gene regulatory networks. *Annu Rev Biomed Eng*, 5:179–206, 2003. doi: 10.1146/annurev.bioeng.5.040202.121553.

M. Kaern, T. C. Elston, W. J. Blake, and J. J. Collins. Stochasticity in gene expression: from theories to phenotypes. *Nat Rev Genet*, 6(6):451–464, Jun 2005. doi: 10.1038/nrg1615.

A. Kamburov, C. Wierling, H. Lehrach, and R. Herwig. Consensuspathdb–a database for integrating human functional interaction networks. *Nucleic Acids Res*, 37(Database issue):D623–D628, Jan 2009. doi: 10.1093/nar/gkn698.

M. Kanehisa, M. Araki, S. Goto, M. Hattori, M. Hirakawa, M. Itoh, T. Katayama, S. Kawashima, S. Okuda, T. Tokimatsu, and Y. Yamanishi. Kegg for linking genomes to life and the environment. *Nucleic Acids Res*, 36(Database issue):D480–D484, Jan 2008. doi: 10.1093/nar/gkm882.

G. Karlebach and R. Shamir. Modelling and analysis of gene regulatory networks. *Nat Rev Mol Cell Biol*, 9(10):770–780, Oct 2008. doi: 10.1038/nrm2503.

N. Kashtan, S. Itzkovitz, R. Milo, and U. Alon. Efficient sampling algorithm for estimating subgraph concentrations and detecting network motifs. *Bioinformatics*, 20(11):1746–1758, Jul 2004. doi: 10.1093/bioinformatics/bth163.

S. A. Kauffman. Metabolic stability and epigenesis in randomly constructed genetic nets. *Journal of Theoretical Biology*, 22(3):437–467, March 1969. doi: 10.1016/0022-5193(69)90015-0.

E. Klipp, R. Herwig, A. Kowald, C. Wierling, and H. Lehrach. *Systems Biology in Practice*. Wiley-VCH, Weinheim, 2005.

M. Koyutürk, A. Grama, and W. Szpankowski. An efficient algorithm for detecting frequent subgraphs in biological networks. *Bioinformatics*, 20 Suppl 1:i200–i207, Aug 2004. doi: 10.1093/bioinformatics/bth919.

K. Kuhn, S. C. Baker, E. Chudin, M.-H. Lieu, S. Oeser, H. Bennett, P. Rigault, D. Barker, T. K. McDaniel, and M. S. Chee. A novel, high-performance random array platform for quantitative gene expression profiling. *Genome Res*, 14(11):2347–2356, Nov 2004. doi: 10.1101/gr.2739104.

W. P. Kuo, T.-K. Jenssen, A. J. Butte, L. Ohno-Machado, and I. S. Kohane. Analysis of matched mrna measurements from two different microarray technologies. *Bioinformatics*, 18(3):405–412, Mar 2002.

J. Lampinen and A. Vehtari. Bayesian approach for neural networks–review and case studies. *Neural Netw*, 14(3):257–274, Apr 2001.

R. Laubenbacher and B. Stigler. A computational algebra approach to the reverse engineering of gene regulatory networks. *J Theor Biol*, 229(4):523–537, Aug 2004. doi: 10.1016/j.jtbi.2004.04.037.

T. I. Lee, N. J. Rinaldi, F. Robert, D. T. Odom, Z. Bar-Joseph, G. K. Gerber, N. M. Hannett, C. T. Harbison, C. M. Thompson, I. Simon, J. Zeitlinger, E. G. Jennings, H. L. Murray, D. B. Gordon, B. Ren, J. J. Wyrick, J.-B. Tagne, T. L. Volkert, E. Fraenkel, D. K. Gifford, and R. A. Young. Transcriptional regulatory networks in Saccharomyces cerevisiae. *Science*, 298(5594):799–804, Oct 2002. doi: 10.1126/science.1075090.

H. Lehrach, R. Dramanac, J. Hoheisel, Z. Larin, G. Lennon, M. P. Monaco, D. Nizetic, G. Zehetner, and A. Poustka. Hybridization fingerprinting in genome mapping and sequencing. In K. E. Davies and S. Tilgman, editors, *Genome Analysis Volume 1: Genetic and Physical Mapping*, pages 39–81. Cold Spring Harbor Laboratory Press, MA, 1990.

B. Lemon and R. Tjian. Orchestrated response: a symphony of transcription factors for gene control. *Genes Dev*, 14(20):2551–2569, Oct 2000.

G. G. Lennon and H. Lehrach. Hybridization analyses of arrayed cdna libraries. *Trends Genet*, 7(10):314–317, Oct 1991.

M. Levine and R. Tjian. Transcription regulation and animal diversity. *Nature*, 424(6945):147–151, Jul 2003. doi: 10.1038/nature01763.

S. Liang, S. Fuhrman, and R. Somogyi. Reveal, a general reverse engineering algorithm for inference of genetic network architectures. *Pac Symp Biocomput*, pages 18–29, 1998.

D. J. Lockhart, H. Dong, M. C. Byrne, M. T. Follettie, M. V. Gallo, M. S. Chee, M. Mittmann, C. Wang, M. Kobayashi, H. Horton, and E. L. Brown. Expression monitoring by hybridization to high-density oligonucleotide arrays. *Nat Biotechnol*, 14(13):1675–1680, Dec 1996. doi: 10.1038/nbt1296-1675.

L. M. Loew and J. C. Schaff. The virtual cell: a software environment for computational cell biology. *Trends Biotechnol*, 19(10):401–406, Oct 2001.

S. Ma, Q. Gong, and H. J. Bohnert. An arabidopsis gene network based on the graphical gaussian model. *Genome Res*, 17(11):1614–1625, Nov 2007. doi: 10.1101/gr.6911207.

S. Mangan and U. Alon. Structure and function of the feed-forward loop network motif. *Proc Natl Acad Sci U S A*, 100(21):11980–11985, Oct 2003. doi: 10.1073/pnas.2133841100.

S. Mangan, A. Zaslaver, and U. Alon. The coherent feedforward loop serves as a sign-sensitive delay element in transcription networks. *J Mol Biol*, 334(2): 197–204, Nov 2003.

W. Marwan, A. Sujatha, and C. Starostzik. Reconstructing the regulatory network controlling commitment and sporulation in physarum polycephalum based on hierarchical petri net modelling and simulation. *J Theor Biol*, 236(4): 349–365, Oct 2005. doi: 10.1016/j.jtbi.2005.03.018.

H. Matsuno, A. Doi, M. Nagasaki, and S. Miyano. Hybrid petri net representation of gene regulatory network. *Pac Symp Biocomput*, pages 341–352, 2000.

L. Matthews, G. Gopinath, M. Gillespie, M. Caudy, D. Croft, B. de Bono, P. Garapati, J. Hemish, H. Hermjakob, B. Jassal, A. Kanapin, S. Lewis, S. Mahajan, B. May, E. Schmidt, I. Vastrik, G. Wu, E. Birney, L. Stein, and P. D'Eustachio. Reactome knowledgebase of human biological pathways and processes. *Nucleic Acids Res*, 37(Database issue):D619–D622, Jan 2009. doi: 10.1093/nar/gkn863.

J. S. Mattick. Rna regulation: a new genetics? *Nat Rev Genet*, 5(4):316–323, Apr 2004. doi: 10.1038/nrg1321.

V. Matys, O. V. Kel-Margoulis, E. Fricke, I. Liebich, S. Land, A. Barre-Dirrie, I. Reuter, D. Chekmenev, M. Krull, K. Hornischer, N. Voss, P. Stegmaier, B. Lewicki-Potapov, H. Saxel, A. E. Kel, and E. Wingender. Transfac and its module transcompel: transcriptional gene regulation in eukaryotes. *Nucleic Acids Res*, 34(Database issue):D108–D110, Jan 2006. doi: 10.1093/nar/gkj143.

F. Mazzocchi. Complexity in biology. exceeding the limits of reductionism and determinism using complexity theory. *EMBO Rep*, 9(1):10–14, Jan 2008. doi: 10.1038/sj.embor.7401147.

H. H. McAdams and A. Arkin. Stochastic mechanisms in gene expression. *Proc Natl Acad Sci U S A*, 94(3):814–819, Feb 1997.

W. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biology*, 5(4):115–133, December 1943. doi: 10.1007/BF02478259.

P. Mendes. Gepasi: a software package for modelling the dynamics, steady states and control of biochemical and other systems. *Comput Appl Biosci*, 9 (5):563–571, Oct 1993.

P. Mendes. Biochemistry by numbers: simulation of biochemical pathways with gepasi 3. *Trends Biochem Sci*, 22(9):361–363, Sep 1997.

P. Mendes and D. Kell. Non-linear optimization of biochemical pathways: applications to metabolic engineering and parameter estimation. *Bioinformatics*, 14(10):869–883, 1998.

P. Mendes, W. Sha, and K. Ye. Artificial gene networks for objective comparison of analysis algorithms. *Bioinformatics*, 19 Suppl 2:II122–II129, Oct 2003.

R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network motifs: simple building blocks of complex networks. *Science*, 298(5594): 824–827, Oct 2002. doi: 10.1126/science.298.5594.824.

J. Monod and F. Jacob. Teleonomic mechanisms in cellular metabolism, growth, and differentiation. *Cold Spring Harb Symp Quant Biol*, 26:389–401, 1961.

P. T. Monteiro, N. D. Mendes, M. C. Teixeira, S. d'Orey, S. Tenreiro, N. P. Mira, H. Pais, A. P. Francisco, A. M. Carvalho, A. B. Lourenço, I. Sá-Correia, A. L. Oliveira, and A. T. Freitas. Yeastract-discoverer: new tools to improve the analysis of transcriptional regulatory associations in saccharomyces cerevisiae. *Nucleic Acids Res*, 36(Database issue):D132–D136, Jan 2008. doi: 10.1093/nar/gkm976.

D. H. Nguyen and P. D'haeseleer. Deciphering principles of transcription regulation in eukaryotic genomes. *Mol Syst Biol*, 2:2006.0012, 2006. doi: 10.1038/msb4100054.

D. T. Odom, N. Zizlsperger, D. B. Gordon, G. W. Bell, N. J. Rinaldi, H. L. Murray, T. L. Volkert, J. Schreiber, P. A. Rolfe, D. K. Gifford, E. Fraenkel, G. I. Bell, and R. A. Young. Control of pancreas and liver gene expression by hnf transcription factors. *Science*, 303(5662):1378–1381, Feb 2004. doi: 10.1126/science.1089769.

E. M. Ozbudak, M. Thattai, I. Kurtser, A. D. Grossman, and A. van Oudenaarden. Regulation of noise in the expression of a single gene. *Nat Genet*, 31(1): 69–73, May 2002. doi: 10.1038/ng869.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference*. Morgan Kaufmann, September 1988. ISBN 1558604790.

C. A. Petri. *Kommunikation mit Automaten*. PhD thesis, Insitut für Instrumentelle Mathematik, Technische Hochschule Darmstadt, Bonn, 1962.

L. Petzold. Automatic selection of methods for solving stiff and nonstiff systems of ordinary differential equations. *siam j. sci. stat. comput.*, 4:136–148, 1983.

J. W. Pinney, D. R. Westhead, and G. A. McConkey. Petri net representations in systems biology. *Biochem Soc Trans*, 31(Pt 6):1513–1515, Dec 2003. doi: 10.1042/.

A. Platt and R. J. Reece. The yeast galactose genetic switch is mediated by the formation of a gal4p-gal80p-gal3p complex. *EMBO J*, 17(14):4086–4091, Jul 1998. doi: 10.1093/emboj/17.14.4086.

A. Poustka, T. Pohl, D. P. Barlow, G. Zehetner, A. Craig, F. Michiels, E. Ehrich, A. M. Frischauf, and H. Lehrach. Molecular approaches to mammalian genetics. *Cold Spring Harb Symp Quant Biol*, 51 Pt 1:131–139, 1986.

N. Przulj, D. G. Corneil, and I. Jurisica. Modeling interactome: scale-free or geometric? *Bioinformatics*, 20(18):3508–3515, Dec 2004. doi: 10.1093/bioinformatics/bth436.

M. Ptashne. *Genetic Switch: Phage Lambda Revisited*. Cold Spring Harbor Laboratory Press, 3 edition edition, April 2004.

J. Quackenbush. Computational analysis of microarray data. *Nat Rev Genet*, 2 (6):418–427, Jun 2001. doi: 10.1038/35076576.

A. Raj, C. S. Peskin, D. Tranchina, D. Y. Vargas, and S. Tyagi. Stochastic mrna synthesis in mammalian cells. *PLoS Biol*, 4(10):e309, Oct 2006. doi: 10.1371/journal.pbio.0040309.

C. Rangel, J. Angus, Z. Ghahramani, M. Lioumi, E. Sotheran, A. Gaiba, D. L. Wild, and F. Falciani. Modeling t-cell activation using gene expression profiling and state-space models. *Bioinformatics*, 20(9):1361–1372, Jun 2004. doi: 10.1093/bioinformatics/bth093.

E. Ravasz, A. L. Somera, D. A. Mongru, Z. N. Oltvai, and A. L. Barabási. Hierarchical organization of modularity in metabolic networks. *Science*, 297 (5586):1551–1555, Aug 2002. doi: 10.1126/science.1073374.

M. H. V. V. Regenmortel. Reductionism and complexity in molecular biology. scientists now have the tools to unravel biological and overcome the limitations of reductionism. *EMBO Rep*, 5(11):1016–1020, Nov 2004. doi: 10.1038/sj.embor.7400284.

B. Ren, F. Robert, J. J. Wyrick, O. Aparicio, E. G. Jennings, I. Simon, J. Zeitlinger, J. Schreiber, N. Hannett, E. Kanin, T. L. Volkert, C. J. Wilson, S. P. Bell, and R. A. Young. Genome-wide location and function of dna binding proteins. *Science*, 290(5500):2306–2309, Dec 2000. doi: 10.1126/science.290.5500.2306.

D. Repsilber and J. T. Kim. Developing and testing methods for microarray data analysis using an artificial life framework. In *ECAL*, pages 686–695, 2003.

F. Rosenblatt. The perception: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

S. Roy, M. Werner-Washburne, and T. Lane. A system for generating transcription regulatory networks with combinatorial control of transcription. *Bioinformatics*, 24(10):1318–1320, May 2008. doi: 10.1093/bioinformatics/btn126.

G. Ruvkun. Molecular biology. glimpses of a tiny rna world. *Science*, 294(5543): 797–799, Oct 2001. doi: 10.1126/science.1066315.

J. Schäfer and K. Strimmer. An empirical bayes approach to inferring large-scale gene association networks. *Bioinformatics*, 21(6):754–764, Mar 2005a. doi: 10.1093/bioinformatics/bti062.

J. Schäfer and K. Strimmer. Learning large-scale graphical gaussian models from genomic data. In J. F. F. Mendes, S. N. Dorogovtsev, A. Povolotsky, F. V. Abreu, and J. G. Oliveira, editors, *Science of Complex Networks: From Biology in the Internet and WWW: CNET 2004*, volume 776, pages 263–276. AIP, 2005b. doi: 10.1063/1.1985393.

J. Schaff, C. C. Fink, B. Slepchenko, J. H. Carson, and L. M. Loew. A general computational framework for modeling cellular structure and function. *Biophys J*, 73(3):1135–1146, Sep 1997. doi: 10.1016/S0006-3495(97)78146-3.

M. Schena, D. Shalon, R. W. Davis, and P. O. Brown. Quantitative monitoring of gene expression patterns with a complementary DNA microarray. *Science*, 270(5235):467–740, Oct 1995.

M. J. Schilstra and H. Bolouri. The Logic of Gene Regulation. In *ICSB '02: Proceedings of the 3rd International Conference on Systems Biology*, pages 197–198, Karolinska Institutet, Stockholm, Sweden, 2002.

M. J. Schilstra and C. L. Nehaniv. Bio-logic: gene expression and the laws of combinatorial logic. *Artif Life*, 14(1):121–133, 2008. doi: 10.1162/artl.2008.14.1.121.

T. Schlitt and A. Brazma. Current approaches to gene regulatory network modelling. *BMC Bioinformatics*, 8 Suppl 6:S9, 2007. doi: 10.1186/1471-2105-8-S6-S9.

G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2): 461–464, 1978.

C. E. Shannon and W. Weaver. *A Mathematical Theory of Communication*. University of Illinois Press, Champaign, IL, USA, 1963. ISBN 0252725484.

S. S. Shen-Orr, R. Milo, S. Mangan, and U. Alon. Network motifs in the transcriptional regulation network of Escherichia coli. *Nat Genet*, 31(1):64–68, May 2002. doi: 10.1038/ng881.

I. Shmulevich, E. R. Dougherty, and W. Zhang. From boolean to probabilistic boolean networks as models of genetic regulatory networks. *Proceedings of the IEEE*, 90(11):1778–1792, 2002.

B. M. Slepchenko, J. C. Schaff, I. Macara, and L. M. Loew. Quantitative cell biology with the virtual cell. *Trends Cell Biol*, 13(11):570–576, Nov 2003.

L. A. Soinov, M. A. Krestyaninova, and A. Brazma. Towards reconstruction of gene networks from expression data by supervised learning. *Genome Biol*, 4 (1):R6, 2003.

P. Sonego, A. Kocsor, and S. Pongor. Roc analysis: applications to the classification of biological sequences and 3d structures. *Brief Bioinform*, 9(3):198–209, May 2008. doi: 10.1093/bib/bbm064.

N. Soranzo, G. Bianconi, and C. Altafini. Comparing association network algorithms for reverse engineering of large-scale gene regulatory networks: synthetic versus real data. *Bioinformatics*, 23(13):1640–1647, Jul 2007. doi: 10.1093/bioinformatics/btm163.

P. T. Spellman, G. Sherlock, M. Q. Zhang, V. R. Iyer, K. Anders, M. B. Eisen, P. O. Brown, D. Botstein, and B. Futcher. Comprehensive identification of cell cycle-regulated genes of the yeast saccharomyces cerevisiae by microarray hybridization. *Mol Biol Cell*, 9(12):3273–3297, Dec 1998.

O. Sporns and R. Kötter. Motifs in brain networks. *PLoS Biol*, 2(11):e369, Nov 2004. doi: 10.1371/journal.pbio.0020369.

M. P. H. Stumpf, C. Wiuf, and R. M. May. Subnets of scale-free networks are not scale-free: sampling properties of networks. *Proc Natl Acad Sci U S A*, 102 (12):4221–4224, Mar 2005. doi: 10.1073/pnas.0501179102.

K. Takahashi, N. Ishikawa, Y. Sadamoto, H. Sasamoto, S. Ohta, A. Shiozawa, F. Miyoshi, Y. Naito, Y. Nakayama, and M. Tomita. E-cell 2: multi-platform e-cell simulation system. *Bioinformatics*, 19(13):1727–1729, Sep 2003.

P. K. Tan, T. J. Downey, E. L. Spitznagel, P. Xu, D. Fu, D. S. Dimitrov, R. A. Lempicki, B. M. Raaka, and M. C. Cam. Evaluation of gene expression measurements from commercial microarray platforms. *Nucleic Acids Res*, 31(19): 5676–5684, Oct 2003.

J. Tegnér and J. Björkegren. Perturbations to uncover gene networks. *Trends Genet*, 23(1):34–41, Jan 2007. doi: 10.1016/j.tig.2006.11.003.

D. Thieffry, A. M. Huerta, E. Pérez-Rueda, and J. Collado-Vides. From specific gene regulation to genomic networks: a global analysis of transcriptional regulation in escherichia coli. *Bioessays*, 20(5):433–440, May 1998. doi: 3.0.CO;2-2.

R. Thomas. Boolean formalization of genetic control circuits. *Journal of Theoretical Biology*, 42(3):563–585, December 1973. doi: 10.1016/0022-5193(73)90247-6.

M. Tomita, K. Hashimoto, K. Takahashi, T. S. Shimizu, Y. Matsuzaki, F. Miyoshi, K. Saito, S. Tanida, K. Yugi, J. C. Venter, and C. A. Hutchison. E-cell: software environment for whole-cell simulation. *Bioinformatics*, 15(1):72–84, Jan 1999.

E. P. van Someren, L. F. Wessels, and M. J. Reinders. Linear modeling of genetic networks from experimental data. *Proc Int Conf Intell Syst Mol Biol*, 8:355–366, 2000.

J. Vohradsky. Neural model of the genetic network. *J Biol Chem*, 276(39):36168–36173, Sep 2001a. doi: 10.1074/jbc.M104391200.

J. Vohradsky. Neural network model of gene expression. *FASEB J*, 15(3):846–54, Mar 2001b. doi: 10.1096/fj.00-0361com.

A. Wagner and D. A. Fell. The small world inside large metabolic networks. *Proc Biol Sci*, 268(1478):1803–1810, Sep 2001. doi: 10.1098/rspb.2001.1711.

M. Wahde and J. Hertz. Coarse-grained reverse engineering of genetic regulatory networks. *Biosystems*, 55(1-3):129–136, Feb 2000.

D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, Jun 1998. doi: 10.1038/30918.

D. C. Weaver, C. T. Workman, and G. D. Stormo. Modeling regulatory networks with weight matrices. *Pac Symp Biocomput*, pages 112–123, 1999.

M. Weber, I. Hellmann, M. B. Stadler, L. Ramos, S. Pääbo, M. Rebhan, and D. Schübeler. Distribution, silencing potential and evolutionary impact of promoter dna methylation in the human genome. *Nat Genet*, 39(4):457–466, Apr 2007. doi: 10.1038/ng1990.

A. S. Weigend, D. E. Rumelhart, and B. A. Huberman. Generalization by weight-elimination with application to forecasting. In *NIPS-3: Proceedings of the 1990 conference on Advances in neural information processing systems 3*, pages 875–882, San Mateo, CA, USA, 1990. Morgan Kaufmann Publishers Inc.

X. Wen, S. Fuhrman, G. S. Michaels, D. B. Carr, S. Smith, J. L. Barker, and R. Somogyi. Large-scale temporal gene expression mapping of central nervous system development. *Proc Natl Acad Sci U S A*, 95(1):334–339, Jan 1998.

P. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis, Harvard University, 1974.

P. J. Werbos. Backpropagation Through Time: What It Does and How to Do It. In *Proceedings of the IEEE*, volume 78, October 1990.

A. V. Werhli and D. Husmeier. Gene regulatory network reconstruction by bayesian integration of prior knowledge and/or different experimental conditions. *J Bioinform Comput Biol*, 6(3):543–572, Jun 2008.

A. V. Werhli, M. Grzegorczyk, and D. Husmeier. Comparative evaluation of reverse engineering gene regulatory networks with relevance networks, graphical Gaussian models and Bayesian networks. *Bioinformatics*, 22(20): 2523–2531, Jul 2006. doi: 10.1093/bioinformatics/btl391.

L. F. Wessels, E. P. van Someren, and M. J. Reinders. A comparison of genetic network models. *Pac Symp Biocomput*, pages 508–519, 2001.

M. L. Whitfield, G. Sherlock, A. J. Saldanha, J. I. Murray, C. A. Ball, K. E. Alexander, J. C. Matese, C. M. Perou, M. M. Hurt, P. O. Brown, and D. Botstein. Identification of genes periodically expressed in the human cell cycle and their expression in tumors. *Mol Biol Cell*, 13(6):1977–2000, Jun 2002. doi: 10.1091/mbc.02-02-0030.

C. Wierling, R. Herwig, and H. Lehrach. Resources, standards and tools for systems biology. *Brief Funct Genomic Proteomic*, 6(3):240–251, Sep 2007. doi: 10.1093/bfgp/elm027.

D. R. Wilson and T. R. Martinez. The general inefficiency of batch training for gradient descent learning. *Neural Netw*, 16(10):1429–1451, Dec 2003. doi: 10.1016/S0893-6080(03)00138-2.

L. Wodicka, H. Dong, M. Mittmann, M. H. Ho, and D. J. Lockhart. Genome-wide expression monitoring in saccharomyces cerevisiae. *Nat Biotechnol*, 15 (13):1359–1367, Dec 1997. doi: 10.1038/nbt1297-1359.

S. Yang and K.-C. Chang. Comparison of score metrics for bayesian network learning. *IEEE Transactions on Systems, Man and Cybernetics, Part A*, 32(3): 419–428, May 2002. doi: 10.1109/TSMCA.2002.803772.

M. K. S. Yeung, J. Tegnér, and J. J. Collins. Reverse engineering gene networks using singular value decomposition and robust regression. *Proc Natl Acad Sci U S A*, 99(9):6163–6168, Apr 2002. doi: 10.1073/pnas.092576199.

J. Yu, V. A. Smith, P. P. Wang, A. J. Hartemink, and E. D. Jarvis. Advances to Bayesian network inference for generating causal networks from observational biological data. *Bioinformatics*, 20(18):3594–603, Dec 2004. doi: 10.1093/bioinformatics/bth448.

J. Yu, J. Xiao, X. Ren, K. Lao, and X. S. Xie. Probing gene expression in live cells, one protein molecule at a time. *Science*, 311(5767):1600–1603, Mar 2006. doi: 10.1126/science.1119623.

A. Zaslaver, A. E. Mayo, R. Rosenberg, P. Bashkin, H. Sberro, M. Tsalyuk, M. G. Surette, and U. Alon. Just-in-time transcription program in metabolic pathways. *Nat Genet*, 36(5):486–491, May 2004. doi: 10.1038/ng1348.

# Selbstständigkeitserklärung

Hiermit versichere ich, die vorliegende Arbeit selbständig und ohne unerlaubte Hilfe angefertigt und alle verwendeten Hilfsmittel und Inhalte aus anderen Quellen als solche kenntlich gemacht zu haben. Desweiteren versichere ich, dass die vorliegende Arbeit nie in dieser Form oder einer anderen Form Gegenstand eines früheren Promotionsverfahrens war.

Berlin, 22. Juni 2009

Hendrik Hache